Data trust framework using blockchain and smart contracts

A dissertation submitted to the
College of Graduate and Postdoctoral Studies
in partial fulfilment of the requirements
for the degree of Doctor of Philosophy
in the Department of Computer science
University of Saskatchewan
Saskatoon

By
Sara Rouhani

Permission to Use

In presenting this dissertation in partial fulfilment of the requirements for a Postgraduate degree from the University of Saskatchewan, I agree that the Libraries of this University may make it freely available for inspection. I further agree that permission for copying of this dissertation in any manner, in whole or in part, for scholarly purposes may be granted by the professor or professors who supervised my dissertation work or, in their absence, by the Head of the Department or the Dean of the College in which my dissertation work was done. It is understood that any copying or publication or use of this dissertation or parts thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to the University of Saskatchewan in any scholarly use which may be made of any material in my dissertation.

Disclaimer

Reference in this dissertation to any specific commercial products, process, or service by trade name, trademark, manufacturer, or otherwise, does not constitute or imply its endorsement, recommendation, or favoring by the University of Saskatchewan. The views and opinions of the author expressed herein do not state or reflect those of the University of Saskatchewan, and shall not be used for advertising or product endorsement purposes.

Requests for permission to copy or to make other uses of materials in this dissertation in whole or part should be addressed to:

Head of the Department of Computer Science
176 Thorvaldson Building, 110 Science Place
University of Saskatchewan
Saskatoon, Saskatchewan S7N 5C9 Canada

OR

Dean
College of Graduate and Postdoctoral Studies
University of Saskatchewan
116 Thorvaldson Building, 110 Science Place
Saskatoon, Saskatchewan S7N 5C9 Canada

Abstract

Lack of trust is the main barrier preventing more widespread data sharing. The lack of transparent and reliable infrastructure for data sharing prevents many data owners from sharing their data. Data trust is a paradigm that facilitates data sharing by forcing data controllers to be transparent about the process of sharing and reusing data.

Blockchain technology has the potential to present the essential properties for creating a practical and secure data trust framework by transforming current auditing practices and automatic enforcement of smart contracts logic without relying on intermediaries to establish trust. Blockchain holds an enormous potential to remove the barriers of traditional centralized applications and propose a distributed and transparent administration by employing the involved parties to maintain consensus on the ledger. Furthermore, smart contracts are a programmable component that provides blockchain with more flexible and powerful capabilities. Recent advances in blockchain platforms toward smart contracts' development have revealed the possibility of implementing blockchain-based applications in various domains, such as health care, supply chain and digital identity.

This dissertation investigates the blockchain's potential to present a framework for data trust. It starts with a comprehensive study of smart contracts as the main component of blockchain for developing decentralized data trust. Interrelated, three decentralized applications that address data sharing and access control problems in various fields, including healthcare data sharing, business process, and physical access control system, have been developed and examined.

In addition, a general-purpose application based on an attribute-based access control model is proposed that can provide trusted auditability required for data sharing and access control systems and, ultimately, a data trust framework. Besides auditing, the system presents a transparency level that both access requesters (data users) and resource owners (data controllers) can benefit from. The proposed solutions have been validated through a use case of independent digital libraries. It also provides a detailed performance analysis of the system implementation. The performance results have been compared based on different consensus mechanisms and databases, indicating the system's high throughput and low latency.

Finally, this dissertation presents an end-to-end data trust framework based on blockchain technology. The proposed framework promotes data trustworthiness by assessing input datasets, effectively managing access control, and presenting data provenance and activity monitoring. A trust assessment model that examines the trustworthiness of input data sets and calculates the trust value is presented. The number of transaction validators is defined adaptively with the trust value. This research provides solutions for both data owners and data users' by ensuring the trustworthiness and quality of the data at origin and transparent and secure usage of the data at the end. A comprehensive experimental study indicates the presented system effectively handles a large number of transactions with low latency.

List of my peer-reviewed publications with contents from this dissertation

- Rouhani, Sara, and Ralph Deters. "Performance analysis of ethereum transactions in private blockchain." In 2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS), pp. 70-74. IEEE, 2017.

- Rouhani, Sara, Vahid Pourheidari, and Ralph Deters. "Physical access control management system based on permissioned blockchain." In 2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), pp. 1078-1083. IEEE, 2018.

- Rouhani, Sara, Luke Butterworth, Adam D. Simmons, Darryl G. Humphery, and Ralph Deters. "MediChain TM: A Secure Decentralized Medical Data Asset Management System." In 2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), pp. 1533-1538. IEEE, 2018.

- Pourheidari, Vahid, Sara Rouhani, and Ralph Deters. "A case study of execution of untrusted business process on permissioned blockchain." In 2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), pp. 1588-1594. IEEE, 2018.

- Rouhani, Sara, and Ralph Deters. "Security, performance, and applications of smart contracts: A systematic survey." IEEE Access 7 (2019): 50759-50779.

- Rouhani, Sara, and Ralph Deters. "Blockchain based access control systems: State of the art and challenges." In IEEE/WIC/ACM International Conference on Web Intelligence, pp. 423-428. 2019.

- Rouhani, Sara, and Ralph Deters. "Distributed Attribute-Based access control system using permissioned blockchain", World Wide Web (2021).

- Rouhani, Sara, and Ralph Deters. "Data trust framework using blockchain technology and adaptive transaction validation", 2021 under review.

Acknowledgements

Firstly, I would like to thank my supervisor, Prof. Ralph Deters, for his support, motivation, guidance, and feedback during my Ph.D. research. I am incredibly grateful for his encouraging insight that has motivated me to dream bigger.

I would like to thank my committee: Prof. Julita Vassileva, Prof. Derek Eager, and Prof. Chris Zhang, as well as the external examiner Prof. Rozita Dara, for their valuable feedback and suggestions on my research. They have all played an important role in polishing my research directions.

I would like to thank Prof. Julita Vassileva again for her guidance, mentorship, and lessons all these years.

I would like to thank Prof. Rui Cruz for his mentorship, guidance and feedback during my collaboration with the Linux Foundation.

I would also like to thank my parents for helping me shape my life with positivity, love and passion. Finally, I am incredibly thankful to my spouse. Without his tremendous understanding, encouragement, and patience in the past few years, it would be impossible for me to complete my study.

Dedicated to: my spouse, Vahid, who always supported me and kept me strong in all good and tough times, and my parents, whose passion and encouragement have always inspired me.

Contents

List of Tables

List of Figures

List of Abbreviations

AAA	Authentication, Authorization, and Accounting
ABAC	Attribute-Based Access Control
ACL	Access Control List
ACM	Access Control Matrix
AD	Access Directory
AM	Attribute Manager
AP	Attribute Provider
API	Application Programming Interface
AS	Application Server
AST	Abstract Syntax Tree
BRBAC BN	Blockchain Role-Based Access Control Business Network
BFT	Byzantine Fault Tolerant
BNA	Business Network Archive
BPMN	Business Process Modelling and Notation
BitML	Bitcoin Modelling Language
CC	Cloud Computing
CFT	Crash Fault-Tolerant
DPoS	Delegated Proof of Stake
DAC	Discretionary Access Control
DHT	Distributed Hash Table
DLT	Distributed Ledger Technology
DTP	Data Trust Portal
ECF	Effectively Callback Free
EHR	Electronic Health Record
EMR	Electronic Medical Record
EOA	Externally Owned accounts
EVM	Ethereum Virtual Machine
XACML	eXtensible Access Control Markup Language
XML	Extensible Markup Language
ID	Identifier
IoT	Internet of Things
IBFT	Istanbul Byzantine Fault Tolerant
JSON	JavaScript Object Notation

MAC	Mandatory Access Control
MSP	Membership Service Provider
PAP	Policy Administration Point
PDP	Policy Decision Point
PEP	Policy Enforcement Point
PIP	Policy Information Point
PRP	Policy Retrieval Point
PBFT	Practical Byzantine Fault Tolerant
PoI	Proof of Importance
PoS	Proof of stake
PoW	Proof of Work
PKI	Public Key Infrastructure
RBAC	Role-Based Access Control
SP	Smart Policy
STM	Software Transactional Memory
TBAC	Transaction Based Access Control
TPS	Transactions Per Second
URI	Uniform Resource Identifier
VM	Virtual Machine
ZKP	Zero Knowledge Proofs

1 Introduction

Trust is still the main obstacle preventing more widespread data sharing. There is a lack of transparent infrastructure to demonstrate how the data will be re-used and for what purposes. This prevents many data owners from sharing their data. Data trust is a new paradigm to improve trust in data stewardship by providing a transparent, reliable, structured and trustworthy framework.

Data trust helps with engaging users to participate in data sharing by increasing the level of trust and decreasing the level of associated risks and uncertainties. Data trust motivates data owners to share their data by pushing data controllers to be transparent about the process of sharing and using data [164, 182].

Data trust addresses the trust issue for not only data providers but also data consumers and data subjects. Data owners require to trust that data users will not misuse their data. Likewise, data consumers need to trust that the data they get access to is trustworthy and sound provenance in order to undertake the delivery of accurate analysis and testify the obtained results [124]. Besides, data consumers need to trust other parties when they share their trained model or analysis results with the data owners or other potential customers eventually.

Sharing personal data might not only put the interests of data subjects at risk but also introduce a new risk to the organization by providing an advantage for its competitors. It is expected to mitigate some of the perceived risks of data sharing and establish trust with a transparent and structured data trust framework.

Kieron O'Hara [124] demonstrates the necessary functions of data trust. He sets out eight technical properties that are essential for trustworthy data sharing: (1) discovery, (2) provenance, (3) access controls, (4) access, (5) identity management, (6) auditing of use, (7) accountability, (8) impact. Many of these essential properties, such as provenance and auditing, are provided by blockchain and its unique inherent features. Some other desirable properties for data trust, such as access control regulations and data impact calculations, can also be programmed into blockchain as smart contracts.

Blockchain is a particular type of distributed ledger technology. The data is recorded on the blockchain as a group of transactions called blocks. Each block has a hash value, and it links to the previous block by referencing the hash value of the previous block. So, the data manipulation is not possible in blockchain because any change leads to an inconsistency that can easily be recognized by the network. In order to attach a valid block to the blockchain, a consensus mechanism applies. There are several consensus mechanisms with a trade-off between performance and security.

Smart contracts are logics encoded in blockchain to create more flexible transactions. They are also able to enforce their conditions automatically using the distributed nature of the blockchain. Before the development

of smart contracts, blockchain applications were limited to creating cryptocurrencies and simple monetary transactions. However, with the development of smart contracts, a new era for blockchain systems began by providing an infrastructure for creating more diverse blockchain-based applications.

Access control refers to any action to prevent data and resources from unauthorized access, disclosure or modification. In traditional databases, an authorization is defined by the triple of $< O, S, P >$, where the subject s is authorized to execute privilege p on object o [34]. There are three main access control policies: 1) Discretionary Access Control (DAC) or authorization-based, 2) Mandatory Access Control (MAC), 3) Role-Based Access Control (RBAC) [148].

In addition, there is another access control model called Attribute-Based Access Control (ABAC). ABAC is an access control model that regulates access permissions, based on the characteristics—in this context called attributes—of subjects, resources, and context (or environment). Access decisions are made by evaluating these attributes based on predefined policies. ABAC is a flexible and fine-grained mechanism that is also capable of enforcing the other three methods.

Blockchain characteristics such as immutability, durability, auditability, and reliability lead to considering blockchain as a supplementary solution for access control systems. Immutability implies that any data on the blockchain can never be changed. Durability means that the data recorded on blockchain can never be lost. Auditibality means that users can trace every transaction, and a tamper-proof audit trail is available. Reliability comes from the decentralized nature of blockchain. Since there is no single point of failure, and the data is stored on many peers, the data can be retrieved at any time with high availability. Also, blockchain applies a consensus mechanism to validate transactions and control malicious behaviours.

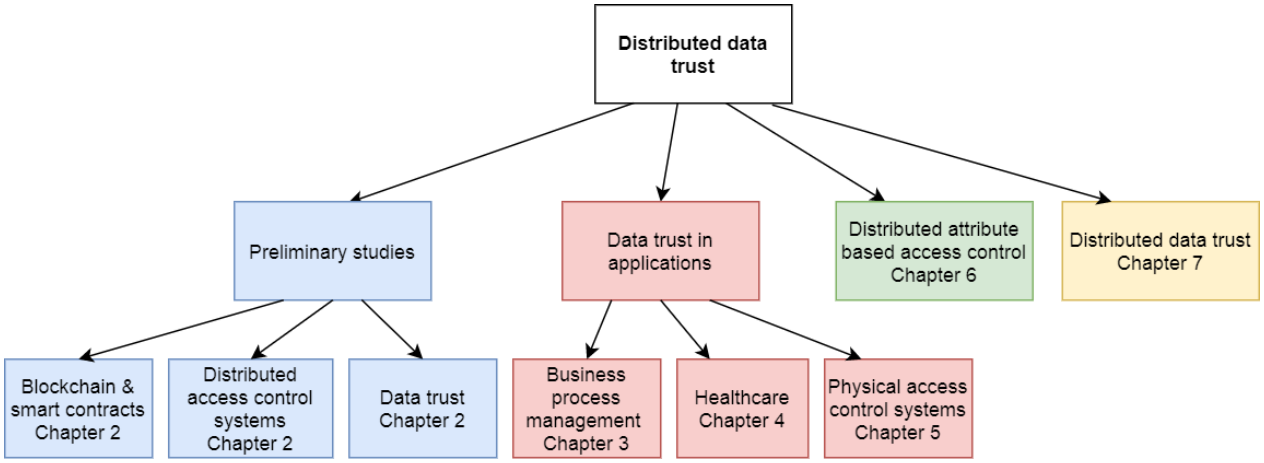

Figure 1.1: Research path.

Figure 1.1 presents the research path followed in this research to study distributed data trust from various angles.

The journey started from preliminary studies related to blockchain and smart contracts. Data trust was investigated in the context of three particular domain-specific applications, healthcare data stewardship [142],

business process management [135], and physical access control systems [145].

Since access control was identified as a critical element in data trust and data sharing applications, the research continued by presenting a distributed ABAC system based on permissioned blockchains. This solution can be employed as an access control system for a variety of applications. The presented solution has been validated through a multi-stakeholder use case of independent digital libraries.

In the end, by utilizing all experience gained from previous studies, this research introduces a distributed and adaptive data trust framework using blockchain technology. This study addresses the concerns of both data owners and data users. Data owners will be equipped with a secure and reliable data-sharing platform with full control over their data access management. Data users also are able to assess the trustworthiness of the data that have been shared with them.

1.1 Motivation

Blockchain is one of the distributed ledger technologies that has fired extensive interest from both academia and industry. Bitcoin [117], as the first application of blockchain, is a cryptocurrency platform. Bitcoin offers a limited scripting language that enables little beyond financial transactions, and its application is limited to simple cryptocurrency transactions.

Smart contracts are programming codes containing arbitrary logic. Smart contracts, as a trusted distributed application, obtain their security from the blockchain and the underlying consensus mechanism. By using smart contracts, it is possible to program more flexible and complex applications and execute them on the blockchain.

Many studies have considered blockchain technology to improve existing services due to its characteristics, such as audibility, immutability, immortality, availability, and smart contracts' functionality. Blockchain has been extensively investigated for digital identity management, access control, data provenance. All these items are essential components for data trust.

This dissertation studies the essential components for data sharing and presents an end to end framework for data trust by utilizing blockchain. It starts by exploring the smallest processing unit of blockchain [143], transactions, and then it examines smart contracts to present an access control system. Ultimately, the presented studies are in the direction of proposing a blockchain-based data trust framework.

I have examined smart contracts design, implementation and limitations. A systematic review of all the topics related to smart contracts, including programming languages, supporting platforms, security aspects, performance aspects, and decentralized applications [140] are presented. Although these applications are studying various fields such as healthcare, IoT, supply chain, digital identity, voting and more, their primary purpose is similar as they aim to control the access over specific data. However, the data domain is their main difference; for example, the application's data could be patients' healthcare data or data generated by Internet of Things (IoT) devices. Therefore, the big picture is utilizing blockchain for access management.

Blockchain has promising features that make it a suitable alternative for access control systems. The distributed nature of blockchain solves many problems of centralized systems. It eliminates third parties, so we do not need to be concerned about privacy leakage from their side. Moreover, we can access a trustable and unmodifiable history log of access requests and access responses for auditing purposes. Consensus mechanisms are applied, so only valid transactions are recorded on the blockchain. Furthermore, we can employ smart contracts to monitor and enforce access permissions flexibly and dynamically.

Although many studies have investigated blockchain for data sharing applications, those are not complete solutions as they focus on a particular aspect such as privacy, or decentralization [201] or they consider a specific domain such as IoT [102] or healthcare [184]. The lack of a complete solution for data trust has motivated me to present an end-to-end data trust framework levering blockchain technology.

1.2 Research questions

The research questions addressed in this thesis center around three main categories based on blockchain technology: data trust, smart contracts and data access control. In the following, the research questions related to these categories are presented.

1.2.1 Smart contracts

Decentralized applications based on blockchain center around smart contract design and implementation. It is important to be familiar with smart contracts' potentials and challenges before developing a new decentralized application using blockchain. In [140], a literature review on smart contracts topic is presented to answer the following questions:

- What are the existing languages and available platform for smart contracts?
- What are the common security problems in smart contracts and what solutions exist to address them?
- How can we improve the performance of smart contracts execution and what tools we have to measure the performance of smart contracts?
- What are the design patterns of smart contracts based on different applications categories?

1.2.2 Access control systems

Access control is one of the essential properties for implementing a system for data trust. Previous research has shown that the primary concern of many represented applications based on blockchain in different domains such as healthcare, IoT, and supply chain is providing an efficient and secure access control mechanism. I have studied the state of the art and challenges of blockchain-based access control systems [144]. This study presents answers to the following research questions.

- What are the problems with current access control systems?
- How blockchain can help to solve these problems?
- What are the challenges for implementing an access control system based on blockchain?
- What are the features of implemented prototype systems?

Understanding the challenges, features, and design patterns of smart contracts, particularly in access control and data sharing applications, has helped me present three access control systems. Two of them target a particular domain, sharing medical (presented in chapter 4) data and physical access control systems (presented in chapter 5), and the last one is a general-purpose ABAC access control system (presented in chapter 6).

1.2.3 Blockchain and data trust

Key characteristics of blockchain such as immutability, security, consensus, and auditing make blockchain an eligible infrastructure for implementing data trust and use it as a platform for sharing data. This dissertation addresses the following questions:

- Why blockchain is a suitable infrastructure for implementing data trust?
- How can we implement data trust's required properties using blockchain?
- How can we develop an end-to-end solution for data trust using blockchain?

1.3 Contribution

The main contributions of this dissertation include the following:

1. I comprehensively and systematically reviewed smart contracts topics from various angles. I reviewed their implementation aspects, security enhancement approaches, performance improvement studies and decentralized application based on smart contracts. The smart contract design, structure, features, and implementation are investigated in those studies. The result of this literature review is published in [140] as a peer-reviewed journal paper.
2. I investigated the problems of centralized and traditional access control systems and discussed how we could address them with blockchain technology. I reviewed the access control studies based on blockchain by presenting the state of the art and future direction. The result of this review paper is published in [144] IEEE/WIC/ACM/ conference.

3. I studied data trust in an untrusted business process management, where multiple untrusted business parties can collaborate for a common purpose in a trusted environment. I investigated the applicability of the order processing execution as a real-word untrusted business process on the permissioned blockchain. I implemented a proof of concept based on Hyperledger Composer to demonstrate blockchain technology's effectiveness for the presented purpose. The result of this study is published in [135] as an IEEE conference paper.

4. I presented a patient-centric system for access management and medical data sharing. The system utilizes blockchain's smart contract to manage access permissions under the control of patients. I implemented a proof of concept to evaluate the proposed solution. This study is one of the earliest implementations of blockchain interactions with off-chain data storage. The result of this study is published in [142] as an IEEE conference paper.

5. I proposed a novel application for employing blockchain in physical access control systems. It is the first study that investigated blockchain effectiveness in improving the security and reliability of software layers in physical access control systems by managing access permissions through smart contracts and audit access requests from the tamper-proof ledger. I implemented the system using Hyperledger Composer and evaluated the performance of the system Using Hyperledger Caliper. The result of this study is published in [145] as an IEEE conference paper.

6. I presented a decentralized attribute-based access control system based on Hyperledger Fabric. This study is the result of my internship with the Linux Foundation, Hyperledger Project. I evaluated the system through a case study of digital libraries. I conducted a comprehensive performance analysis, which indicated that the system could carry out hundreds of access decisions with an average latency of 0.54 seconds. I evaluated the presented system based on two different orderer (consensus) services, Raft and Kafka, two databases, CouchDB and GoLevelDB, and various network configurations. The evaluation results indicate that the proposed system effectively handles a throughput of hundreds of access decisions transactions per second, with an average latency of 0.54 seconds per transaction.

7. Relying on the foundation built from previous studies, I introduced an end-to-end data trust framework with adaptive transaction validations. This study presents a trust model based on three factors: confidence, endorsement, and reputation. I present an access control and consent management service for data owners to share their data assets and a data provenance service for monitor and audit access requests and permissions. I implemented the system using Hyperledger Fabric permissioned blockchain. The evaluation result shows that the blockchain-based system can effectively calculate trust value for 2502 data assets in 500 seconds with an average latency of 0.37 seconds.

1.4 Summary of chapters

- Chapter 1 (Introduction): This chapter includes motivations of the dissertation, problem statement and research questions that are answered in this dissertation, and contribution of presented studies.
- Chapter 2 (Background and related works): This chapter introduces the background of blockchain systems, smart contracts, access control systems, and data trust. It also reviews the related work regarding blockchain-based access control systems, decentralized applications based on smart contracts, and data trust systems.
- Chapter 3 (Establish trust in business process management): This chapter discusses permissioned blockchain advantages for executing an untrusted business process. The implementation of the order processing scenario on permissioned blockchain is discussed and evaluated.
- Chapter 4 (Sharing medical data using blockchain): This chapter presents an architecture and implementation of a patient-centric medical data management system. The system is developed based on permissioned blockchain to control medical data access and off-chain data storage.
- Chapter 5 (Physical access control system): This chapter introduces blockchain applicability for the software management layer in physical access control systems. The system architecture, implementation and evaluation are discussed.
- Chapter 6 (Decentralized attribute-based access control system): This chapter discusses the system model, architecture, and implementation of the ABAC model on Hyperledger Fabric. The system evaluation and performance analysis results are included in this chapter.
- Chapter 7 (Data trust framework): This chapter presents an end-to-end data trust framework. The framework includes data asset trustworthy assessment and trust evaluation, as well as access control and consent management. The system implementation details and performance evaluation results are presented at the end of this chapter.
- Chapter 8 (Conclusion and future direction): This chapter includes a summary of the dissertation, main contributions and future directions.

2 Background and Related works

This chapter reviews the research studies in the areas of smart contracts, decentralized access control systems based on smart contracts and blockchain, and data trust initial concepts and related studies.

The results of these studies are published in two separate review papers. In the first study, I conducted a comprehensive review of smart contracts [140]. And in the later study, I reviewed the topic of decentralized access control systems based on blockchain and smart contracts [144] with presenting state of the art and the current solutions' research gaps. I present these studies in the following sections.

2.1 Blockchain

Blockchain is a class of distributed ledger technology. The data is recorded on the blockchain as a group of transactions called blocks. Each block has a hash value, and it links recursively to the previous blocks all the way to the first block by referencing the hash value of the previous block in the header of the current block. As a result, data manipulation is not possible in the blockchain, as any change leads to an inconsistency between hashes, which can be easily recognized by the network. In addition to blockchain, there are other data structures in distributed ledger technology, such as directed acyclic graphs (DAG) presented by Tangle [134].

Blockchain applies a consensus mechanism to attach a valid block to the blockchain. There are several consensus mechanisms with a trade-off between performance and scalability [175].

Before developing smart contracts, blockchain applications were limited to generating cryptocurrencies and simple transactions for transferring cryptocurrencies. Smart contracts evolution has provided the foundation for developing more diverse blockchain-based decentralized applications. Smart contracts are logics encoded in blockchain to implement programmable transactions, and they can enforce their conditions automatically.

There are numerous blockchain platforms, which are different in several features. System developers can select a specific platform based on their system requirements, such as the type of network (public or permissioned), supporting language for smart contracts, development complexity, performance, and cost. Currently, blockchain platforms work independently; however, a new generation of blockchain systems is progressing to provide interoperability between multiple blockchain platforms to extend the decentralized applications [18].

Blockchain networks divide into two main categories: public and permissioned. Public blockchains are

open to the world, where everyone can join the blockchain with an anonymous identity, submit a transaction, and participate in consensus. Public blockchains require a heavy computation consensus mechanism to maintain a distributed ledger at a large scale. Permissioned blockchains include an additional membership layer, so only authenticated users can join, and each user can have different access levels. They are more appropriate for enterprise sectors, and they can employ lighter consensus mechanisms because of initial filtering and semi-trusted members.

2.1.1 Blockchain transactions

A transaction is a logical unit of sequential operations, which leads to the state change in database systems. Database systems are classified into two groups based on their transactions' properties: ACID and BASE. ACID databases guarantee four properties for their transactions, Atomicity, Consistency, Isolation, and Durability.

- Atomicity means that either all of the transaction's operations must be completed or not at all.
- Consistency means that the system's integrity must be maintained, and the transactions must change the current valid state of the database to another valid state.
- Isolation means that if transactions execute concurrently, they must produce the same result as they execute sequentially.
- Durability guarantees that committed transactions will remain permanently.

BASE is mainly designed for NoSQL databases and distributed systems, and it is softer than the ACID model. BASE includes three principles, Basic Availability, Soft state, and Eventual consistency.

- Basic Availability means that the system provides essential availability of data in case of partial failure of nodes.
- Soft state refers to the system's state that can be changed during the time, even without data input or update.
- Eventual consistency implies that the BASE systems do not guarantee a specific time for consistency, but eventually, they will reach consistency, unlike ACID systems, which emphasize immediate consistency.

Blockchain transactional properties do not exactly match with neither the ACID nor the BASE models. Tai et al. [169] present a new transaction perspective for blockchains called "SALT", which includes four characteristics, Sequential, Agreed, Ledgered, and Tamper-resistant. Sequential states that blockchain processes transactions sequentially. Agreed refers to the consensus mechanism applied in the blockchain for validating the transactions. In the blockchain, the majority of the nodes must accept a transaction in order to be committed to the system. Ledgered refers to the durability characteristic of the blockchain. Once

a transaction is committed to the blockchain, no one can revoke it. However, as the paper mentions, the blockchain's ledger property is slightly weaker than the durability property of the ACID systems because of the possibility of the fork. Tamper-resident indicates the tamper-proof characteristic of transactions in blockchain, which means that once a transaction is committed to the blockchain, it is impossible to alter it.

2.1.2 Public and permissioned blockchain

Public blockchains are introduced by Bitcoin, which the nodes are untrusted. In addition to public blockchains, there are also permissioned or private blockchains.

Public or permissionless blockchains are open to the world. Everybody with anonymous identity can join the public blockchain, input transactions, and participate in the consensus process. The computational power in public blockchains significantly increases as the number of blocks and the total size of data grows. Currently, most public blockchains use a category of PoW consensus mechanism, which is explained in the next subsection.

Permissioned or private blockchains work similar to the public blockchains, but there is a membership layer on top to authenticate users before joining blockchain. In permissioned blockchains, users could have different access levels for submitting the transactions, reading the transactions, or participating in the consensus mechanism. Because of the initial user filtering, permissioned blockchains can use lighter consensus mechanisms (semi-centralized consensus methods [176]), so they process transactions faster. Hyperledger Fabric [13] is a well-known example of the permissioned blockchains.

2.1.3 Blockchain platforms

Bitcoin [1] is the first application utilized the blockchain. It has mainly developed to manage and transfer Bitcoin (its cryptocurrency). After Bitcoin, developing smart contracts by *Ethereum* [2] led to a new generation of blockchain systems along with various applications.

Today many blockchain platforms are geared toward implementing smart contracts and decentralized applications. Blockchain platforms are emerging and are nearly indistinguishable in some cases from core blockchain technology. However, they are different in various aspects, such as the type of the network(public or permissioned), support for built-in cryptocurrency, transaction workflow, performance, privacy, cost and, most importantly, maturity. Some blockchain platforms such as Ethereum, Hyperledger Fabric, Corda, and Quorum have mature tools, while others offer very little support for their users and developers.

The following overviews the leading blockchain platforms that support smart contracts. Table 2.1 compares these platforms based on different features, including consensus approach, network type, smart contracts programming language and whether they present their cryptocurrency or not.

[1] https://bitcoin.org/en

[2] https://www.ethereum.org/

Table 2.1: Blockchain platforms

Platforms	Consensus	Public or Permissioned	Supports Smart Contracts	Smart Contracts Language	Built-in Cryptocurrency
Bitcoin	PoW	Public	Yes (with support of side chains)	Ivy[3], RSK[4], BitML [16]	Yes (Bitcoin)
Ethereum	PoW and PoS	Both	Yes	Solidity, Flint[151], SCILLA [155]	Yes (Ether)
Hyperledger Fabric [5]	PBFT	Permissioned	Yes	Go, Node.js, Java	No
Neo [6]	DBFT	Both	Yes	C#, VB.Net, F#, Java, Kotlin, Python	Yes (NEO)
Nem [7]	Proof of importance	Both	Yes	Provide template	Yes (XEM)
Quorum [8]	Raft-based and Istanbul BFT	Permissioned	Yes	Solidity	No
Cardano [9]	PoS	Public	Yes	Plutus (Functional Language)	Yes (ADA)
EOS[10]	DPoS	Public	Yes	C++	Yes (EOS)
R3 Corda[11]	Flexible plug-in feature for consensus	Permissioned	Yes	Kotlin	No
Tendermint[12]	BFT	Permissioned	Yes	Any Language	No
Waves[13]	LPoS	Public	Yes	RIDE	Yes (Waves)

After investigating the Ethereum platform and experimenting performance of Ethereum transactions in my first study [143]. I have used Hyperledger Fabric platform for implementing the research conducted in this dissertation. I chose Hyperledger Fabric as the most mature permissioned blockchain, and it was an excellent fit for my proposed applications. In the following, the initial concepts regarding these two platforms are explained.

Ethereum

Ethereum has been designed to be adaptable and flexible with Turing complete built-in scripting language for smart contracts called Solidity. Application developers can develop decentralized applications using smart contracts that are run by the blockchain network. This platform's goal is to facilitate transactions between individuals who would have no means to trust one another.

Ethereum's basic unit is account. Everyone who wants to submit transactions to the blockchain requires an account. Ethereum includes two types of accounts: Externally Owned accounts (EOA) and Contract Accounts. With EOA users send transactions directly, Contract Accounts compose the address and information of smart contracts stored on the blockchain, and smart contracts regulate them.

Every account is defined by a pair of keys, a private key, and a public key. Each account's address comes from the last 20 bytes of the public key, and it is the necessary part of each transaction.

It is essential to understand the difference between transactions and internal transactions. The sender of the transaction uses EOA's private key to sign the transaction, and after confirmation, a hash value returns and using this hash value, it is possible to track it. However, there is no signature field for internal transactions. Different sources may use the terms of call or message for internal transactions.

In Ethereum, the transaction is a single instruction code that sends a message from an Externally Owned Account. The ledger launches with a genesis block, and then new transactions process and create new blocks and new states. Any change in the blockchain state starts with a transaction that is sent by EOA. This transaction either directly transfers Ether (Ethereum digital currency) to another account, or it could trigger a smart contract execution. Every transaction includes several fields, and the miner in the network prioritizes the transactions based on the GasPrice field. If multiple transactions have been sent from the same account, the miner calculates them by nonce value. The nonce field in the transaction is equal to the number of the operations sent by the sender of the transaction; this value increments by sending every new transaction.

[3]https://docs.ivy-lang.org/bitcoin/
[4]https://www.rsk.co/
[5]https://www.hyperledger.org/projects/fabric
[6]https://neo.org
[7]https://nem.io/
[8]https://www.jpmorgan.com/global/Quorum
[9]https://cardanodocs.com/introduction/
[10]https://eos.io/
[11]https://www.r3.com/corda-platform/
[12]https://tendermint.com/
[13]https://wavesplatform.com/

Therefore, the transactions sent from the same account with the same gas price will be executed, respectively [181]. Figure 2.1 illustrates the diagram of transaction processing in Ethereum platform.

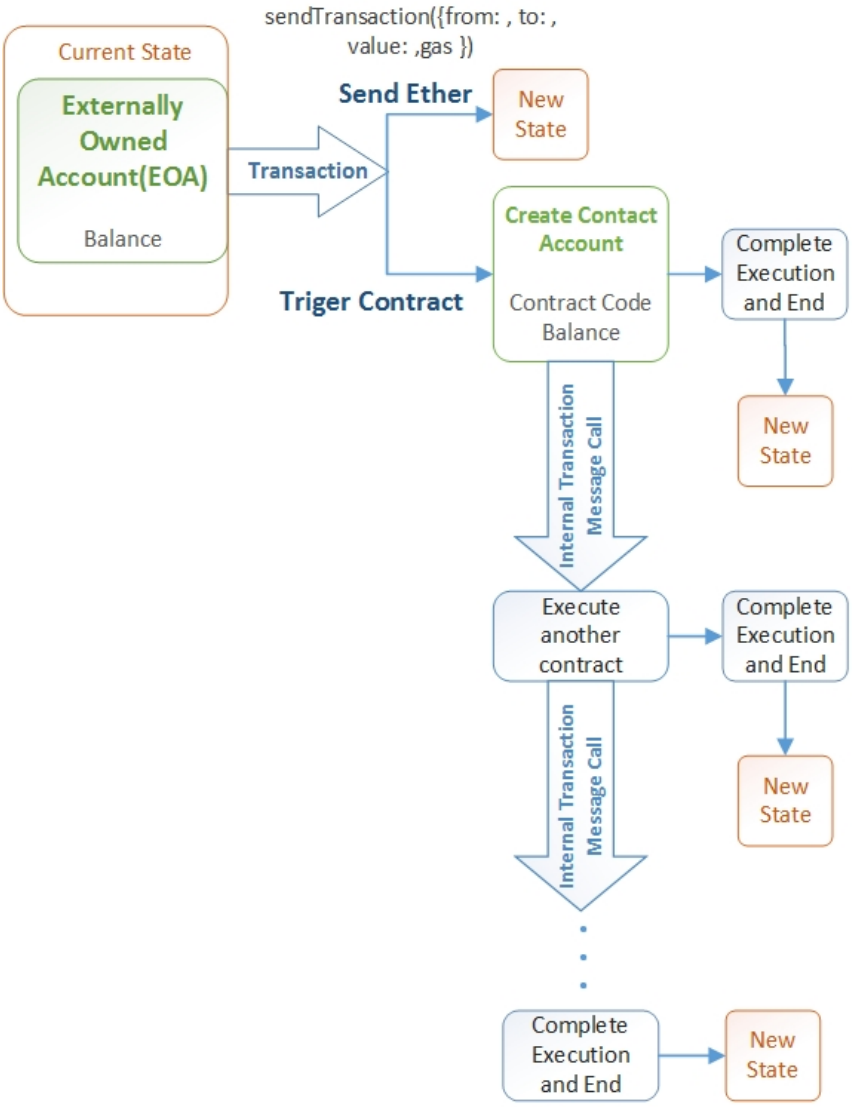

Figure 2.1: Ethereum transaction processing diagram [143].

We can consider Ethereum as a transaction-based state machine. Ethereum runs a state transition function to ensure that the current state's transition would lead to a new valid state. Formula 2.1 shows the formal explanation of a valid state transition based on Ethereum Yellow paper [181]. γ is the Ethereum state transition function, and T represents the transaction.

$$\sigma_{t+1} = \gamma(\sigma_t, T) \tag{2.1}$$

Ethereum and many other blockchain platforms such as Tendermint and Quorum follow the order-execute architecture for processing transactions. There are several drawbacks to this approach [13] as following.

- Consensus is hard-coded within the platform, so it is impossible to change the consensus method without high expenses.
- The consensus protocol determines the transaction's trust model; therefore, it cannot be adapted based on smart contract implementation.
- Smart contracts must be written in a domain-specific language, and using generic programming languages for smart contracts cause problems because they cannot ensure deterministic results.
- Transactions must be deterministic.
- Every smart contract runs on all peers, which can raise issue and prohibits the dissemination of contract code and state to a subset of peers.
- Most importantly, transactions execute consecutively, which affects performance drastically.

Hyperledger Fabric

Hyperledger Fabric [13] is a permissioned blockchain hosted by the Linux Foundation. Hyperledger Fabric has a modular structure that allows component pluggability, such as consensus, membership, and database. The membership layer can authenticate users and grant users access based on their access level and system policy.

Hyperledger Fabric is very mature in terms of being permissioned as we can configure its network so that users have different and granular access levels. For example, we can configure a network that only a subset of users could participate in consensus, submit a transaction, run a particular smart contract, and read the ledger's state.

Hyperledger Fabric smart contracts and chaincode

Hyperledger Fabric introduces the terms of *chaincode*, a program code that implements the application logic and runs during the execution phase. In Hyperledger Fabric, the terms smart contract and chaincode are used interchangeably. However, a smart contract indeed represents the transaction logic and then it is packaged as a chaincode to be deployed to the Fabric blockchain network. A chaincode could include one or multiple smart contracts, and the underlying of each smart contract is a set of transaction definitions. When we deploy a chaincode to the network, all smart contracts within it are made available to applications. Figure 2.2 illustrates an example of access control chaincode.

Supporting general-purpose programming languages such as Java, Go, and JavaScript for writing smart contracts (chaincodes) in Hyperledger Fabric indulges many blockchain developers and researchers. There are two types of chaincode in Hyperledger Fabric. The first is similar to typical smart contracts written by untrusted developers, and they can write their own logic and execute it on the blockchain. The second chaincode type is called *system chaincodes*. They manage the blockchain system and maintain parameters.

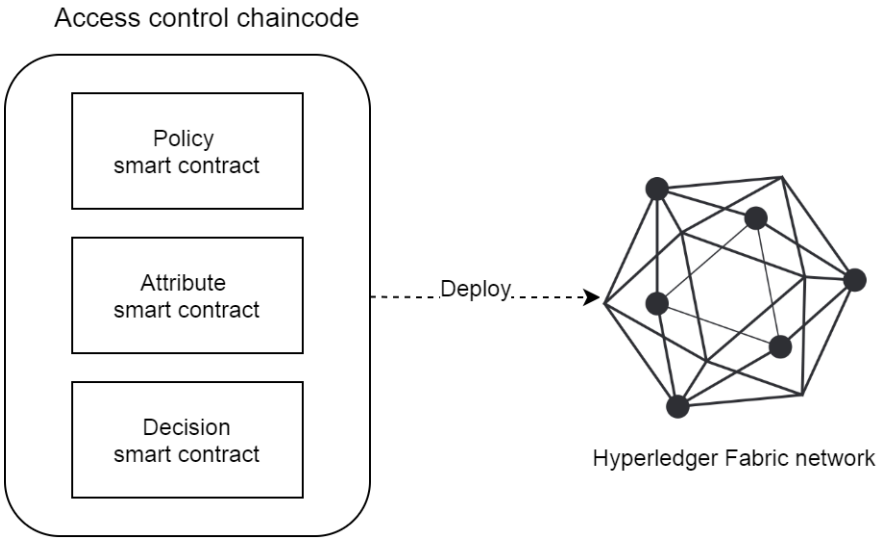

Figure 2.2: Hyperledger Fabric chaincode example.

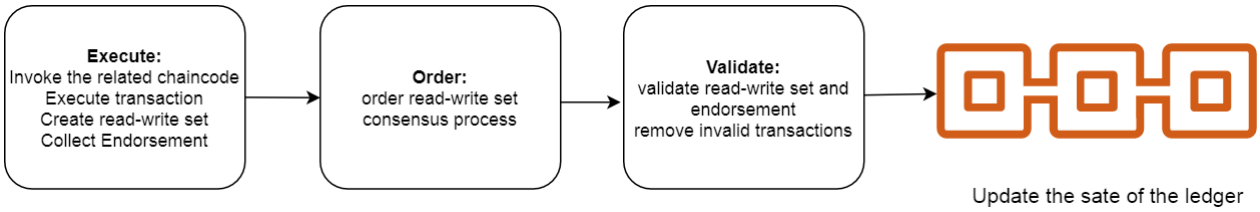

Figure 2.3: Hyperldger Fabric execute-order-validate architecture.

Hyperledger Fabric transaction processing

Hyperledger Fabric introduces a new architecture for transaction flow called execute-order-validate architecture that separates transaction execution from consensus and enables policy-based endorsement. This novel architecture is designed to address the issues mentioned in the previous section regarding order-execute architecture. In the execution phase, the executing peers execute transactions, check them against correctness, create read-write sets. In the order phase, read-write sets are ordered via consensus protocol or called *orderer* unit. Finally, in the validation phase, transactions are checked based on endorsement policy, and the endorsing peer will validate read-write sets again. Endorser peers evaluate all transactions within the block in parallel.

After the validation phase, Each peer appends the block of ordered transactions to the Fabric's chain, and the write sets are committed to the current state database for every valid transaction. Figure 2.3 illustrates the execute-order-validate architecture.

Hyperledger Fabric transaction validation and endorsement

There is an endorsement policy with every deployed chaincode that determines which organizations in a blockchain network must sign a transaction generated by a given smart contract included in the current

chaincode. The transaction is indicated as valid if all required organizations specified in the endorsement policy sign the transaction. The endorsement unit is responsible for collecting the signatures needed for each transaction and validating transactions. Fabric introduces a syntax for specifying customizable endorsement policies. Endorsement policies are expressed in terms of identities matched to a role.

Endorsement policies are what make Hyperledger Fabric different from other blockchain platforms such as Ethereum or Bitcoin. In these systems, any node in the network can submit valid transactions. Hyperledger Fabric more realistically models the real world; trusted organizations in a network must validate transactions.

The Hyperledger Fabric has also introduced a key-level or state-based endorsement policy, which can specify more granular policies at the level of the keys stored on the ledger. It means for every key-value entity; we can specify a distinct endorsement policy.

In addition to collecting signatures from all required organizations specified in the endorsement policy, the endorsing peer nodes must check the transaction to ensure that the world state's current value matches the transaction's read set and then sign it. This step is specified to make sure that there has been no intermediate update. If a transaction passes both these checks, then it is marked as a valid transaction.

Hyperledger Fabric membership service

Hyperledger Fabric is a permissioned blockchain. In Fabric, every participant or actor requires an identity encapsulated in an X.509 digital certificate. The actor could be a peer, orderer, client application, or system administrator. This identity includes attributes indicating the actor's permission level to access the different information and components in the Fabric network.

Fabric CA[14] is a certificate authority that issues identities by generating a public and private key, which forms a key-pair that can be used to authenticate identity. Fabric Membership Service Provider (MSP) verifies the identities issued by Fabric CA without revealing the actor's private key by maintaining a list of permissioned identities. It also determines the identities' roles. In Fabric, every actor has particular privileges by specifying their roles. There are various roles in Hyperledger Fabric, such as admin, peer, client, and orderer.

2.1.4 Consensus mechanisms

The consensus mechanism concludes the validating process of transactions in blocks and reaching an agreement between all peers. Numerous consensus mechanisms are available, which are different in computation power, performance, scalability, and tolerating disruptive behaviours. Wang et al. [176] studied blockchain consensus algorithms focusing on both distributed consensus system design and incentive mechanism design for public blockchains. Nguyen and Kim also [120] conducted a survey on consensus algorithms. They classify the consensus mechanisms into two categories: proof-based and voting-based. In the proof-based method, one

[14]https://hyperledger-fabric-ca.readthedocs.io/

leader (or multiple leaders) is selected, and the leader is responsible for validating and attaching blocks to the ledger. Their primary difference is how they can choose the leader. In the voting-based approach, multiple nodes vote for every block validation and based on consensus policy, minimum positive votes are required for block validation. In the following, the consensus mechanisms, which mainly applied by the blockchain platforms mentioned in table 2.1 are introduced.

Proof of Work (PoW) [61] is a consensus mechanism proposed by S. Nakamoto for Bitcoin and then later applied by Ethereum. Proof of work is an incentive-based method that miner nodes must solve a complicated mathematical puzzle in order to gain rewards. The process is like constantly guessing until they solve the puzzle and find a value called the nonce. The first miner that unlocks the puzzle is the winner of the current block. Then the block is announced to other nodes for verification. The verification process is not a complicated task like the original puzzle. Then nodes check the proposed block against cheatings, and then the collection of ordered transactions will be committed as a new block to the blockchain. PoW is very resilient against tampering, but it requires very high computation and energy power.

Proof of stake (PoS) originally introduced by Peercoin [15] to reduce the cost of PoW. PoS is based on the proof of ownership of cryptocurrency. In every round, the network chose a miner based on the nodes' stake amounts. As much as a node is wealthier, the chance to be selected is more. The miner gets rewarded by proposing a correct block after validating it by other nodes. PoS reduces the amount of computation, but there is a possible problem that the rich nodes get richer every time. It means wealthier nodes would have a better opportunity to get selected each time.

Delegated Proof of Stake (DPoS) [153] is similar to PoS, but only a subset of nodes (witness) can participate in the block generation process. Stakeholders select the block generator members. Simply any node that holds any amount of token is considered as a stakeholder. Because the number of validator nodes is less, DPoS is faster and more efficient.

Proof of Importance (PoI) has been introduced by the NEM blockchain platform [3]. The network assigns a rating to each account according to its importance using graph theory methods. Accounts with higher importance have a higher chance to append a new block to the ledger. As much as an account uses blockchain and transfers coins, its importance rate increases. PoI doesn't require high power.

Practical Byzantine Fault Tolerant (PBFT) [35] is based on the Byzantine Fault Tolerant consensus. First, nodes select a leader. The leader is not constant, and the nodes replace the leader frequently. Each round includes three phases: pre-prepared, prepared, and commit for adding a new block to the blockchain. In the pre-prepared phase, the leader orders the transactions and proposes a new block as a proposal to the remaining nodes. In the preparation phase, the other nodes announce their votes to the leader and other nodes. Eventually, if only two-thirds of the nodes accept the proposal, a new block will be accepted and committed to the blockchain. Figure 2.4 shows the three phases in PBFT. PBFT can tolerate less than 33 malicious nodes, so it is suitable for permissioned blockchain, as the members must be authenticated before

[15] https://peercoin.net/

joining the network.

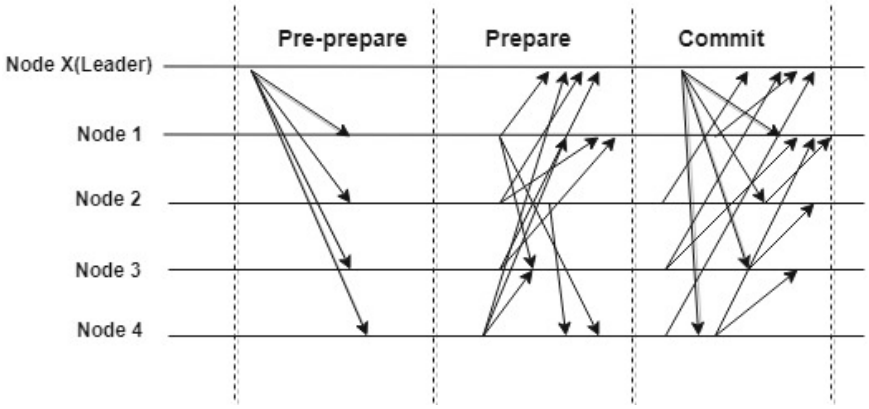

Figure 2.4: PBFT three handshaking phases [35].

Raft [125] is a Crash Fault Tolerant (CFT) consensus mechanism used by Quorum, R3 Corda, and Hyperledger Fabric platforms. Raft is a leader-follower method. There are three states in Raft: follower, candidate, and leader. The procedure starts from the follower state. After that exact times out, if a follower does not receive a response from the leader, it shifts to the candidate state. In the candidate state, nodes vote for the next leader. The candidate that receives maximum votes will be selected as the next leader; otherwise, it remains in the election state or reverts to the follower state. If the leader discovers a server with a higher term in the leader state, the leader leaves the leader state and reverts to the follower state.

Istanbul Byzantine Fault Tolerant (IBFT) [2] is stimulated by PBFT [35] with some modifications. Similar to PBFT, IBFT is a three-phase approach, pre-prepared, prepare, and commit. The system can tolerate F faulty nodes if we have N validators where N= 3F+1. IBFT is a final protocol. It means a fork is not possible in IBFT. In IBFT, there is no client to send the proposed block, and every validator can offer a suggesting block. Other validators select one validator in each round, and the elected validator broadcasts a new block (pre-prepared message). Then validators that received pre-prepare message broadcast prepare message. If validators receive 2F+1 prepare messages, they enter the preparation phase and then if they accept the proposed block, they announce the commit message. If validators receive 2F+1 commit messages and they begin the commit phase, the block will be attached to the blockchain. Istanbul BFT has been employed by Quorum blockchain.

2.1.5 Trust model of blockchain platforms

In general, a network of mutually untrusting peers maintains the blockchain ledger. Transactions are validated through a consensus protocol, and any changes apply in all the ledger's copies. In this way, we reach distributed trust among a set of untrusted peers, and we can execute trusted distributed applications on top of the blockchain.

However, the level of the untrustworthiness of peers is different in public and permissioned blockchains.

In public blockchains, all participants are entirely untrusted, and for security and maintaining trust in the network, complex and high computational consensus mechanisms, such as PoW, are required. In contrast, in permissioned blockchains, the parties do not fully trust each other. However, because of the initial filtering, the involved nodes are semi-trusted, so that a lighter consensus protocol, such as traditional Byzantine-Fault Tolerant (BFT), is enough to achieve security and trust.

Most of the permissioned blockchain platforms still use a fixed trust model because they rely on asynchronous BFT replication mechanisms to establish consensus [173]. Such protocols operate based on a security assumption that among $n > 3f$ peers, up to f are tolerated to misbehave, which are called Byzantine faults. However, in permissioned blockchains, it is possible to separate the trust model of transaction validation from the consensus protocol. The trust model can be adapted to the smart contract requirements so that trust assumptions based on the application requirements can be customized. Hyperledger Fabric implemented these flexible trust assumptions as a part of the transaction endorsement policy. Based on the required trust level in a smart contract and involving parties, the number of peers required to endorse (confirm) transactions can be tailored [13].

2.2 Smart contracts

Smart contracts are programs that execute on the blockchain. With the development of blockchain platforms that support Turing-complete language for smart developing smart contracts, we now have more flexibility to develop various transactions with the more complex procedure and utilize them for multiple application domains. The blockchain can automatically enforce smart contracts' terms and conditions. Each smart contract can include one or multiple transactions. A smart contract can be deployed using a transaction that consists of the code for a set of functions and the smart contract's initial state.

This section reviews the smart contracts related studies from multiple angles. It starts with discussing smart contracts' implementation in terms of available programming languages for writing smart contracts and the code location design in different blockchain platforms. Then it reviews the studies that aim to improve the security and performance of smart contacts. Finally, it presents multiple decentralized applications based on blockchain and smart contracts.

This section provide answer to the following questions:

- What are the existing languages and available platform for smart contracts?
- What are the common security problems in smart contracts and what solutions exist to address them?
- How can we improve the performance of smart contracts execution and what tools we have to measure the performance of smart contracts?
- What are the design patterns of smart contracts based on different applications categories?

Table 2.2: Smart contracts programming languages.

Platform	Script language	Turing complete	Computation level	Virtualization
Bitcoin	Script	No	Low-level (Virtual Machine)	No
Ethereum	EVM bytecode	Yes	Both Low-level and High-level (Ethereum Virtual Machine) (Solidity and serpent)	No
Nxt	-	No	High-level (JavaScript)	No
Hyperledger Fabric	-	Yes	High-level (Golang, node.js, Java)	Yes (Docker)

2.2.1 Smart contracts programming language

Some blockchain platforms such as Hyperledger Fabric, Neo, EoS, and Tendermint support general programming languages such as Java, C++, NodeJS, Python, and Go for smart contracts, and some other platforms, such as Ethereum, presented languages particularly for writing smart contracts, such as Solidity. In this section, we overview the programming languages designed mainly for developing smart contracts. Table 2.1 specifies the supporting platforms for these smart contracts languages. In general, there are two main trends for developing a new programming language for smart contracts. First, using logic and specifics to make smart contracts more straightforward to understand and more similar to traditional contracts. The second studies aim to develop a programming language to improve the security of smart contracts.

Seijas et al. [154] also review the scripting languages used in three blockchain systems: Bitcoin, Ethereum and Nxt. Table 2.2 shows the summary and the specification of designated smart contracts scripting languages in addition to Hyperledger Fabric that we added for completion. In the following, I present the list of programming languages mainly designed for writing smart contracts.

Solidity

Solidity [49] is the most popular programming language particularly developed for writing smart contracts. Solidity is an object-oriented, Turing complete language, and it runs on Ethereum Virtual Machine (EVM). Ethereum is not the only blockchain platform that supports Solidity; Quorum, Hyperledger Burrow, and Hyperledger Besu also use Solidity. Even platforms such as Hyperldger Fabric, which is designed to promote general programming languages (Node, Java, and Go) for writing smart contracts, has also integrated Solidity smart contracts [207]. Solidity is the most well-known and mature contract-oriented language. However, it is vulnerable against security attacks [27]. Therefore, many researchers propose improvements in Solidity smart contracts' security by introducing mid-level languages or smart contracts analyzing tools.

Logic-based smart contracts

Idelberger et al. [83] suggested logic-based or declarative language as a replacement of procedural languages to make smart contracts more related to traditional contracts. They state the advantages of logic-based smart contracts as follow:

- They are more reasonable for contract parties.
- The logic-based smart contracts can embed contract parties' negotiations easier.
- They can be validated easier.
- In terms of storage, logic-based contracts are lighter.
- They can ease the interoperability between contracts by utilizing rule interchange languages.
- The process of contract alteration is easier.

First, the paper explored a Pseudo-code of the contractual licensing clauses as an instance of a typical contract. Then it represented the formal contract logic implemented by defeasible logic engine SPINdle [96]. This study proposes two possible solutions for implementing logic-based smart contracts, off-chain and on-chain implementation. There is a centralized system at the top of the blockchain in the off-chain implementation, and a centralized server executes smart contracts. In the on-chain option, smart contracts can form and negotiate off-chain or on-chain, including the following steps: contract storage, enforcement and monitoring, and regulations, which should be executed on-chain and stored on the blockchain. The authors identified obvious challenges resulting from the implementation of logic-based smart contracts, asking further studies to implement a more efficient and cheaper logic-based algorithm.

SPESC (a specification language for smart contracts)

SPESC is a specification language for defining the specs of smart contracts [74]. SPESC creates an abstraction contract at the top of the smart contract written by another programming language. The developers designed SPESC to make smart contracts understandable for diverse collaboratives required in smart contracts design, such as business experts and lawyers, like logic-based smart contracts [83]. It presents smart contracts in a similar way as real-world contracts by providing the specifications that describe each party, a set of contract terms, commitments and rights of the parties, and the contract circumstances. SPESC smart contracts include four elements: contract parties, properties, terms (include obligations and rights as subclasses), and data type definitions (include the primitive type and complex type as subclasses). SPESC supports multiple expressions to define term conditions. Figure 2.5 shows SPESC supported expressions and transactions. The paper also explores the understandability of SPESC by running an experiment according to questionnaire results of over fifteen participants from computer science and law departments. The results demonstrate that SPESC is more understandable than Solidity smart contracts; however, the paper nor translate SPESC to

Solidity or any other smart contract programs nighter deploy it on a real blockchain platform to investigate possible challenges.

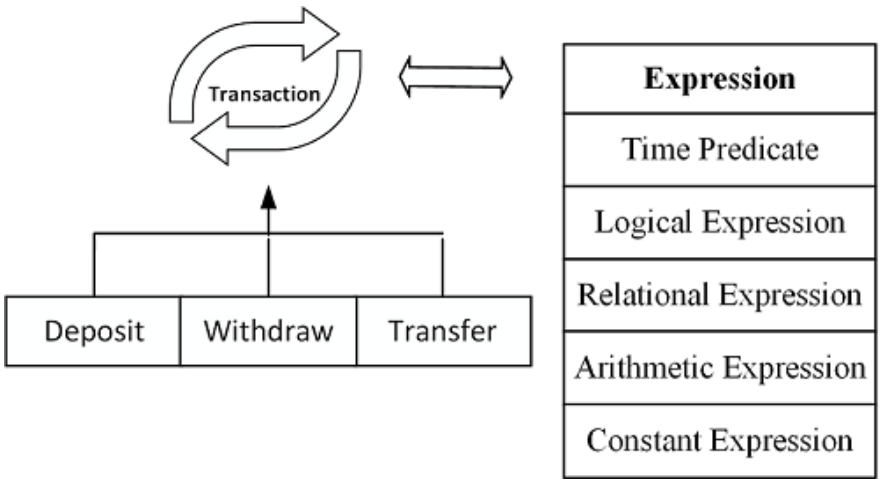

Figure 2.5: SPESC supported expressions and transactions.

Bamboo

To solve the problem of reentrancy (it is explained in section 2.2.3) in Solidity smart contracts, Yoichi Hirai introduced a programming language to create polymorphic contracts for Ethereum called Bamboo [77]. It forces the state-machine method to programmers. Hirai inspired Bamboo's syntax from Erlang. The following code shows three contracts written in Bamboo that they deploy together. Contract A can become C or abort, and contract C can become D, and the program forces their orders.

```
Contract A() {
        case (bool A'){
        // Do something
        Return (true) then become A();
}
case (bool A''()){
        if(something)
                return (true) then become
                C();
        else
                abort;
        }
}
Contract C(){
        case(void C'()){
        if (something)
```

```
                //Do something
        else
                return then become D();
        }
 }
Contract D(){
        //Do something
}
```

SCILLA (intermediate-level language)

SCILLA [155] is an intermediate-level language for smart contracts that use formal methods to verify the smart contract written in higher-level languages such as Solidity. They structure smart contracts as communicating automata to examine smart contracts' semantics, security, and consistency properties. SCILLA achieves expressivity and traceability by presenting smart contracts in communicating automata and separating programming concepts. SCILLA implies a separation between computation and communication, separation between effectual and pure computation, and separation between invocation and continuation.

Flint

Flint [151] is a domain-specific statically-typed programming language for smart contracts, which addresses security. Flint includes caller capabilities blocks to check the access permission of Ethereum accounts and contract functions. Flint introduces protection blocks to add a security layer to limit smart contract execution and protect smart contracts from unauthorized calls. Flint does not permit the creation, duplication, and destruction of assets, but they can be split, merged or transferred. Flint also categorizes smart contracts' functions into two groups: mutating function, which changes the contract state and non-mutating functions. In Flint, invoking the mutating functions by non-mutating functions is restricted, and it is not possible.

BitML (Bitcoin Modelling Language)

BitML [16] is a high-level programming language for Bitcoin smart contracts to define the terms of trading Bitcoins. It generates smart contracts in symbolic expressions (symbolic model) and then compiles these expressions to Bitcoin scripts. Symbolic expressions can be interpreted easier by using formal methods. Also, they argue that any violation at the computation level is also identifiable at the symbolic level. BitML can implement many Bitcoin smart contract-based services such as escrow services, timed commitments, lotteries, and gambling games. However, it does not support express contingent payments (smart contracts that enable selling solutions for a class of NP problems).

2.2.2 Smart contracts code location

Smart contracts are programs that execute logic, and like transactions, they can change the ledger's state. They can automatically execute and enforce their terms without the intervention of trusted authorities. Smart contract codes are written in a high-level programming language. The smart contract code's location is dependent on the platform. For example, they can store on the ledger like transactions or install on network peers.

In the Ethereum platform, smart contracts are in the *Solidity* language, and Solidity codes compile into an Ethereum binary format called EVM (Ethereum Virtual Machine) bytecode. Any smart contract resides at a specific address in EVM bytecode. Any referral to the smart contract is through invoking the address of the smart contract. It means smart contracts' bytecode is a part of the ledger in Ethereum platform.

Hyperledger Fabric introduces a new term for smart contracts called chaincode. A chaincode is a package of related smart contracts; however, it could include only one smart contract, so chaincode and smart contract terms are used interchangeably. In Hyperledger Fabric, one peer installs it's proposed chaincode. The system administrators instantiates the chaincode to its channel (the channel is a private blockchain overlay in Hyperledger Fabric which allows private communication between multiple organizations, the Fabric network includes at least one channel) to make chaincode available for all peers belong to the same channel and make sure that all peers use the same version of the chaincode. Every peer that wants to submit a transaction or query data through chaincode must also install the chaincode. The program calls chaincodes in Hyperledger Fabric through their names and versions.

2.2.3 Smart contracts security

Smart contracts might handle a large sum of money, digital assets, stocks, or data. Considering smart contracts' security is essential considering even a small bug can lead to critical problems, such as a significant amount of money loss or privacy leakage. For example, Ethereum well-known crowdfunding smart contract, *DAO (Decentralized Autonomous Organization)*, was attacked in June 2016 because of a bug in its code and resulted in 60 million USD loss [29]. The attacker exploited the reentrancy vulnerability. The vulnerable program recursively called the split DAO function to transfer Ether (Ethereum cryptocurrency) to the attacker account, and the calls stopped before updating the new balance of the calling contract (the attacker contract). Writing secure and bug-free smart contracts is a difficult task [52] as previous studies show that a high percentage of smart contacts that already are deployed on the Ethereum blockchain are vulnerable. Luu et al. investigate 19,366 Ethereum Smart contracts using their represented tool called OYENTE, and their report determines that more than 45 percent of them are buggy [104].

This section outlines represented methods and tools to address the security issues in smart contracts. Table 2.3 shows a summary of described strategies, which address smart contracts' security.

Table 2.3: Smart contracts security methods.

Represented Methods	References
Formal verification	F* [20][168], Isabelle/HOL [10], KEVM [131][76], solidity* & EVM* [20]
Smart contracts code analyzer tools	OYENTE [104], EthIR [7], GasTap [8] SmartInspect [23], SECURIFY [172], MAIAN [121], Vandal [25], Smartcheck [171], GASPER [38]
Effective callback	[64]
Using control flow	[60]
New consensus method	[177]
Securely connect blockchain to off-chain	Town Crier [196], Smart Cast [94]
Privacy	Hawk [93], Enigma [208]
Secure programming language	SCILLA [156], Flint [152]

Classification of security problems

Luu et al. [104] proposed classes of security problems in Ethereum smart contracts, and they offer methods to refine the operational semantic of Ethereum to enhance the security of smart contracts. They also developed *OYENTE* as a symbolic execution tool, which can identify relevant bugs in Ethereum smart contracts. The smart contracts vulnerabilities are as follow:

Transaction Ordering Dependence (TOD): In the Ethereum blockchain, the order of transaction execution is up to miners, and clients do not control them. This problem happens when the same contract invokes more than one transaction, and the order of the transactions can affect the new state of the blockchain. The authors recommend guard condition as a solution. The idea is that the transaction confirmation becomes dependent on the guard condition satisfaction; otherwise, the transaction is declined.

Timestamp Dependence: This problem is relevant to smart contracts, which include conditions that trigger by block timestamp. Block timestamps are set by miners based on their local system time, and so it can be manipulated by an adversary. The authors suggest using block index instead of block timestamp because it is incremental and protected from manipulation.

Mishandled Exceptions: This problem targets the smart contracts that call another smart contract. If any exception occurs in a called contract, it terminates and returns false, but it may not inform the caller of a smart contract. The paper suggestion for this problem is adding explicit throw and catch EVM instructions.

Reentrancy Vulnerability: This problem is related to DAO vulnerability (it was explained at the beginning of section 2.2.3). When a contract calls another contract, the current contract execution waits until the called contract termination. It provides an opportunity for the adversary to exploit the caller contract intermediary state and call its methods numerous times.

Smart contracts security analysis tools

OYENTE: OYENTE [16] is a tool to analyze Ethereum smart contracts code based on symbolic execution [104]. OYENTE takes two inputs: Ethereum smart contract bytecode and Ethereum global state. It checks smart contracts against four previously declared problems in the previous section. There are four main elements in OYENTE: CGFBuilder, Explorer, CoreAnalysis, and Validator. CGFBuilder is responsible for creating a Control Flow Graph of the contract. Explorer symbolically executes contracts. The output of Explorer sends to CoreAnalysis to check the existence of four main problems. Validator removes false positives to make sure OYENTE reports detailed problems to the user. OYENTE's report declares that more than 45 percent of the Ethereum contracts include at least one of the four indicated problems based on analyzing 19,366 contracts.

EthIR: EthIR is an extension of OYENTE. EthIR analyzes the Ethereum Bytecodes instead of the source code of smart contracts. Therefore, it is helpful in analyzing smart contracts that their source code is not available. It acts as a decompiler that modified CFGs provided by OYENTE to create a rule-based representation (RBR) of the bytecodes that fit high-level analyses [7].

SmartInspect: Once the developer deploys the smart contract, it is hard to inspect the given contract attributes because of smart contracts data's encoded nature. Bragagnolo et al. addressed verifying deployed smart contracts [23]. SmartInspect is a tool for analyzing deployed smart contracts using decompilation techniques, and mirror-based [22] reflection for remotely deployed on a reflection-less system [130] without redeploying the contact (getter method) or using an API to accumulate raw data (ad-hoc decoding method). SmartInspect first parses contract codes to generate AST (Abstract Syntax Tree). Then by interpreting the structured representation of AST, it produces the mirror. The next step uses the mirror to extract contracts data, then data is given into four different formats: 1) REST 2) Pharo widget user interface 3) JSON 4) HTML. The authors evaluate the represented tool by comparing SmartInspect with getter and ad-hoc decoder methods in nine different features include four characteristics represented in [130]: interactiveness, distribution, security, and instrumentations in addition to important blockchain features: privacy, pluggability, consistency, reusability, and unrestricted types. Table 2.4 shows the evaluation results.

SECURIFY: Ttsankov et al. developed SECURIFY [172], another smart contract security platform. SECURIFY first decompiles the EVM bytecode of the smart contract and derives semantic facts that are data and control flow dependencies. Finally, it checks the security pattern, which is represented in domain-specific language. SECURIFY patterns include a set of compliance and violation—one out of the three labels of

[16] https://github.com/melonproject/oyente

Table 2.4: SmartInspect evaluation result [23].

Characteristic	SmartInspect	Getter	Decoder
Interactiveness	Yes	Partial	Partial
Distribution	Yes	No	No
Security	Yes	Yes	Yes
Instrumentation	No	No	No
Privacy	Yes	No	Yes
Pluggability	Yes	No	Yes
Consistency	Yes	Yes	Yes
Reusability	Yes	No	No
Unrestricted Types	Yes	No	Yes

Compliance, Violence, and Warning mark smart contracts unsafe behaviours. If the semantic fact matches the compliance pattern, it considers compliance; else, it is considered violence if it fits violence. If it matches neither of those patterns, it is recognized as a warning behaviour.

MAIAN: MAIAN is an analysis tool represented by Nikolic et al. [121] to detect greedy, prodigal, or suicidal behaviour of Ethereum smart contracts based on a trace of vulnerabilities. Greedy smart contracts refer to smart contracts that stay alive but lock Ether endlessly. Prodigal smart contracts do not provide any backup solution in case of attacks, and they can leak Ether to an arbitrary address. Suicidal smart contracts are those, which can be destroyed by any arbitrary account by forcing the smart contract to commit suicide. Their analysis on 970,898 smart contracts detected 34,200 (2.365 distinct) vulnerable contracts.

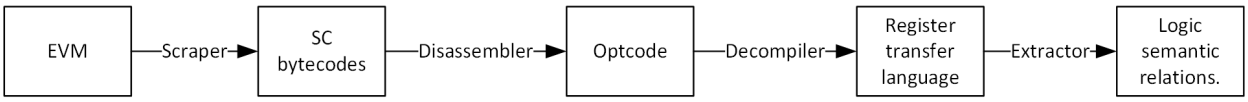

Figure 2.6: Vandal's pipeline [25].

Vandal: Brent et al. introduced Vandal as a static security analysis framework for smart contracts [25]. Vandal uses an analysis pipeline to translate smart contracts bytecodes to logic relations, reflecting the smart contracts' program semantics. The analysis pipeline includes a bytecode scraper, a disassembler, a decompiler, and an extractor. Figure 2.6 shows Vandal pipeline. Also, the authors employed Souffle' [85], a declarative language, as a datalog engine to express security vulnerabilities. This study denotes five vulnerabilities: Unchecked send, Reentrancy, Unsecured Balance, Destroyable contract, and Use of Origin.

SmartCheck: Tikhomirov et al. classified common Solidity code issues into four classes: Security, Functional, Operational, and Development [171]. They deployed SmartCheck that translates smart contracts' source code (Solidity code) to the XML format, and it looks for problematic patterns using XPath queries.

Table 2.5: SmartCheck smart contract code issue classification [171].

Security issue	Functional issue	Operational issue	Development issue
Balance equality	Integer division	Byte array	Token API violation
Unchecked external call	Locked money	Costly loop	Compiler version not fixed
DoS by external contract	Unchecked math		Private modifier
Send instead of transfer	Timestamp dependence		Redundant fallback function
Reentrancy	Unsafe type inference		Style guide violation
Malicious libraries			Implicit visibility level
Using tx.origin			

Table 2.5 presents the SmartCheck code issues classification.

Table 2.6: Under optimized patterns [38].

Category	Under-optimized Pattern
Useless code related	1. Dead Code: Existing bytecodes in smart contracts, which is never executed 2. Opaque Predicates: The presence of the statement operations in smart contracts, which their results are same.
Loop Related	3. Expensive Operations: Moving expensive operation outside of the loop can lead to saving a significant amount of gas. 4. Constant Outcome: If the result of the loop outcome is always constant that loop can be removed. 5. Loop Fusion: Combining several loops into one loop. 6. Repeated computations: Loops can have constant result in each iteration. This pattern saves gas by computing the outcome once and then reusing it. 7. Comparison with the unilateral outcome: Comparisons that always result the same value in the loop. They do not recognize in compilation such as pattern two.

GASPER: Gas is a unit introduced by the Ethereum blockchain used for the execution fee that the transactions' sender or smart contracts should pay for every operation. In other words, gas is the fee to reward miners for executing the program. Gas can be exchanged with Ether, which is Ethereum cryptocurrency [28]. Chen et al. [38] explain a security vulnerability in smart contracts that leads to unnecessary gas consumption. They propose seven under-optimized patterns in smart contracts classified into two groups: useless-code related and loop-related. Table 2.6 shows the seven under-optimized smart contracts patterns based on Solidity programming language. They also developed a tool called GASPER for identifying overcharged three out of seven representative patterns automatically: dead code, opaque predicates, and expensive operations in a loop. The result of scanning 4,240 smart contracts shows that at least 80% of contracts involved in one of these three patterns and at least 73.2% of smart contracts suffer from two out of three patterns, and 71.7% of smart contracts suffer from all three patterns.

Formal verification

Formal verification is one of the most precise methods to verify the system's accuracy. It is also utilized to verify the behaviour of smart contracts. This section reviews the studies that identify and solve smart contracts' security vulnerabilities using formal methods.

Bhargavan et al. [20] introduce a framework to analyze and verify smart contracts written in Solidity programming. The authors used F* [168], a functional programming language for program verification. They introduce two tools: Solidity*, a tool that translates Solidity programs to F* codes for verifying the source level functional accuracy and EVM*, a tool which is a decompiler for EVM bytecodes to perform a stack analysis and generates equivalent F* programs for verifying low-level properties. The tool verifies that both outputs are equivalent. The output of Solidity* indicates the dangerous patterns to detect send() function failure exception and reentrancy. The output of EVM* checks the limit of gas consumption in smart contracts methods.

Amani et al. [10] similarly use the decompilation technique to verify Ethereum smart contracts at the bytecode level by employing logical framework Isabelle/HOL. They define the smart contracts' correctness features by relying on the Ethereum termination guarantee gas concept. They separate smart contracts bytecode into the basic blocks and formulate a sound program logic for verification.

Park et al. [131] present another formal verification tool to verify EVM bytecodes by adopting KEVM [76], which is a comprehensive executable formal semantics of the EVM. They denote a group of challenges, including Byte-Manipulation Operations, Arithmetic Overflow, Gas Limit, and Hash Collision, to verify EVM bytecodes and propose some techniques to address them.

Detection of effective callback free objects

Grossman et al. [64] discuss a safety notion called ECF (Effectively Callback Free) and specifically DECF (Dynamically ECF) for executions and SECF (Statically ECF) for objects. Based on the represented definition "an execution of an object is DECF when there exists an equivalent execution of the contract without callbacks, which starts in the same state and reaches the same final state. An object is considered SECF when all its possible executions are dynamically ECF". ECF is a bug detection factor. It shows that DAO and other buggy contracts are not ECF. Besides, we can use ECF to enable modular reasoning about objects with an encapsulated state. The study presents an online polynomial time and space algorithm to check the execution of smart contracts based on ECF factor and have integrated it into EVM (Ethereum Virtual Machine) [181] with low runtime overhead. This algorithm detects conflict memory accesses and investigates cumulatively of operations to check the accuracy of conditions. The evaluation result indicates that most non-ECF executions are responsible for vulnerable executions, and there are few bugless non-ECF smart contracts.

Control flow

Fröwis et al. [60] expose the mutability problem in smart contracts control flow in Ethereum blockchain. The paper argues that smart contracts control flow is not immutable and has the potential to modify. This study initially discusses how the call graph is generated from Ethereum smart contracts to solve this problem. Then it introduces a clean-up strategy based on a call graph to prevent biases arising from attacks against

Ethereum. Eventually, it measures the trustless smart contracts.

Here are the presented steps to obtain the required content to analyze call relationships in Ethereum smart contracts and create a call graph:

- First, they extracted Ethereum smart contracts bytecode through JSON_RPC API[17].
- Second, to extract the smart contract's code, they iterated all transactions again and selected those with no recipient. This technique is limited to smart contracts created by user accounts, not those generated by code accounts.
- Third, they used tracing mode in parity client[18] to reach internal transactions.
- Fourth, they stored the bytecode, creation block number, and destruction block number of all extracted smart contracts into MongoDB.
- Fifth, they used evmdis[19] to disassemble the EVM bytecodes and achieve reaching definition to identify the source of call addresses.
- At the end, the call graph has been generated from extracted calls. The study considers a smart contract trustless "if and only if all calls in its dependency tree have hardcoded addresses, hence all code that a smart contract can execute is fixed upon deployment of smart contract". They measured trustless smart contracts before and after smart contracts clean-up affected by widespread attacks DAO and DOS (Denial of Service). The results show that 54 percent of active smart contracts were trustless before clean-up based on the represented definition. After clean-up, the ratio increases to 62 percent, which means that two out of five smart contracts deployed on Ethereum need trust in at least one third-party.

Secure consensus method

Watanabe et al. [177] suggest a new consensus method to secure blockchain applied to smart contracts. This study directs the problem of the collapse of coins in proof of stake consensus mechanism. The represented method is based on the collapse of credibility. Credibility is a metric to determine trustable smart contract owners. The credibility of the contract owner is relevant to how many smart contracts the user owns. If a smart contract owner attacks blockchain, it loses trust in the whole society. However, using only the credibility factor increases the chance for fake credibility score and 51% attack. Using hybrid blockchain based on both proof of stack and credibility factor is recommended to solve both problems. Figure2.7 displays using the hybrid method in block mining. Based on the hybrid approach, the two consecutive blocks should not use the same consensus method.

[17]https://github.com/ethereum/wiki/wiki/JSON-RPC
[18]https://www.parity.io/
[19]https://github.com/Arachnid/evmdis/

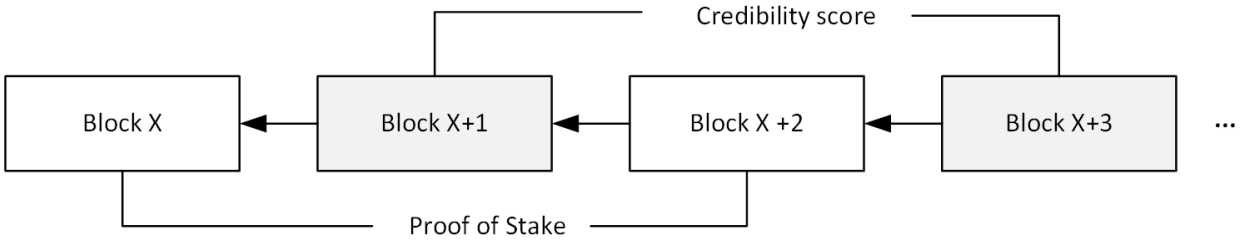

Figure 2.7: Using the hybrid method based on both PoS and credibility score [177].

Connecting smart contracts and off-chain

Town Crier [196] is one of the earliest studies that examine connecting smart contracts to an external resource (off-chain) to request the concise pieces of data called datagrams. Town Crier is an authenticated data feed system that acts as a bridge between Ethereum smart contracts and HTTPS-enabled data sources combined with a trusted hardware backend. The trusted and secure architecture model includes the Town Crier contract, Enclave, and Relay. The Town Crier contract acts as a frontend for the system. It provides an API for the contract that uses the Town Crier service. The Enclave is an instance of the TC core program running in the Trusted Execution Environment, such as SGX enclave and queries data from an HTTPS data source. The Relay is a typical user-space application, and it has provided to relay network traffic between smart contract, Enclave, and HTTPS data source. Figure 2.8 shows TC architecture. The User contract requests a datagram from TC Contract. Relay unit relays the Enclave and Data Source request and finally sends the user contract's response through TC Contract.

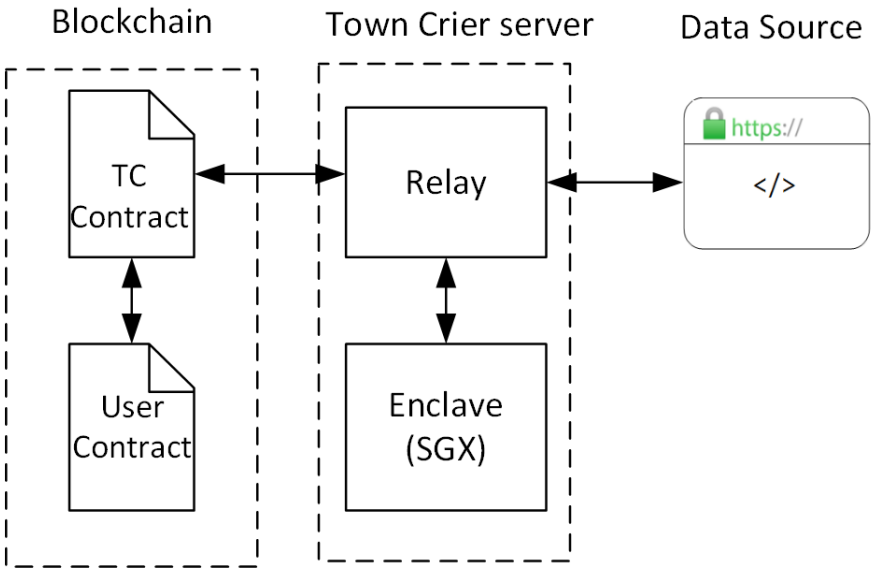

Figure 2.8: Town Crier architecture [196].

Kothapalli et al. [94] represent SmartCast, an incentive-compatible consensus protocol that has adapted

the work of Clement [44] to smart contracts. The main idea is to employ smart contracts to provide incentive compatibility for off-chain consensus protocol. The system financially rewards unbiased parties for consensus and direct participation. The protocol decreases communication with the Ethereum blockchain, so the transaction cost also gets minimized. Since communication is the major cost, the paper outlines this as incentive-compatibility. The smart contract model's essential two components are: 1) a smart contract program, which receives information from nodes about each other's behaviour and recognizes lazy nodes based on that payout reward to each party, 2) local program for each of the parties to execute, which includes interaction with smart contracts and an off-chain consensus process.

Molina-Jimenez et al. [114] propose requirements for obtaining an effective hybrid method of off-chain and on-blockchain solutions. The hybrid model unites decentralized smart contracts and trusted centralized third parties to address the performance and scalability issues in pure blockchain approaches. In the following, I present the practical cases based on the integration of the centralized and decentralized approach:

1. Indelible blockchain-based log: the blockchain part can record passive logs that are worth keeping and duplicating in the blockchain.

2. Cryptocurrency-based payment channel: Using public blockchain, such as Bitcoin or Ethereum, for payment process for a notable amount of money transformation, thus the transaction fee is negligible.

3. Off-blockchain execution of operations: To increase the application's performance and prevent transaction throughput limitation, the study recommended executing primary operations off-chain.

Privacy

This section outlines the studies that approach privacy issues in smart contracts data or related applications. Hawk [93] is a framework for generating privacy-preserving smart contracts. Hawk gets a smart contract, and it's compiler automatically generates the cryptographic protocol. A Hawk contract includes two parts: private and public. The private part includes parties' input data and currency units. The private parts are cryptographically invisible from the public view. The rest of the codes, which are not private, are presumed public. Hawk divides programs into three pieces, and three separated entities execute each one of them. First, the piece that all consensus nodes execute, second, the piece that users execute, and third, the piece the manager executes, which is a minimally trusted party and can see users' private data. Hawk guarantees privacy as long as the manager does not reveal the private portion. Figure 2.9 shows the general overview of the Hawk framework. The paper also provides a formal UC-based (Universal Composability) model of cryptography to examine the represented protocol's security and can be adopted by other decentralized protocols.

Enigma [208] is a decentralized computation network based on Ethereum that provides privacy and computation to allow different parties to store and share secret data. They have developed a Turing-complete scripting language to build private smart contracts that support private data. The Enigma off-chain platform manages private data blockchain and executes the public parts. The system comprises three distributed

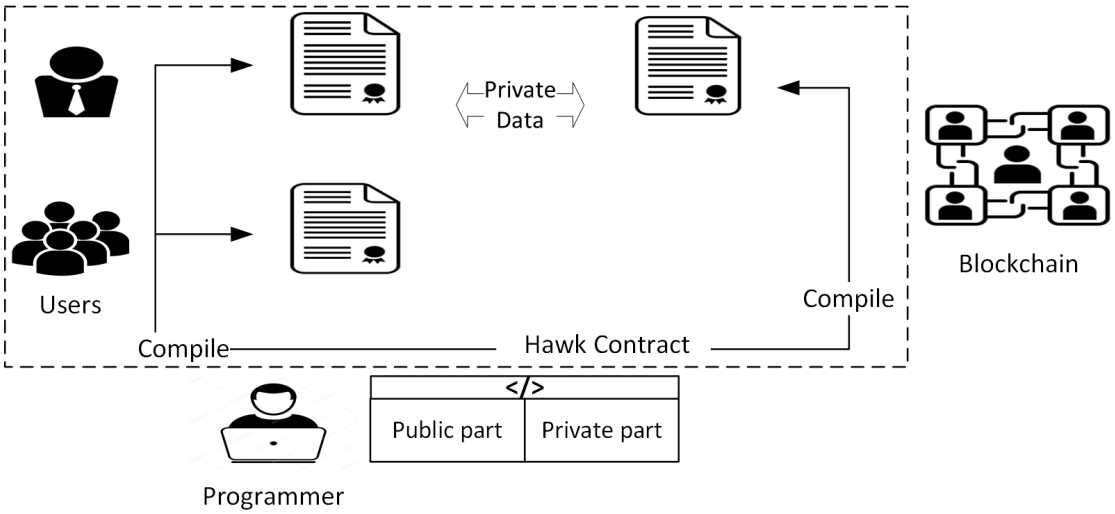

Figure 2.9: Hawk platform architecture [93].

databases. 1) a blockchain to control the system, manage access control, and as a tamper-proof database of events, 2) a Distributed Hash Table (DHT) for recording off-chain private data, 3) a Multi-Party Computation (MPC) to break data into meaningless chunks and spread it between different nodes without replication.

Ekiden [41] also utilizes TEE (Trusted Execution Environment) to achieve privacy and confidentiality for sensitive data associated with smart contracts besides high-performance execution. Ekiden presents a proof of publication method to ensure that the SGX enclaves have been synchronized with the blockchain's latest state.

2.2.4 Smart contracts performance

The performance of executing transactions and smart contracts is a severe challenge in blockchain-based systems, which prevents them from competing with current applications on a large scale. Researchers aim to promote the performance of running smart contracts by presenting methods to running non-conflicting smart contracts transactions concurrently [14, 53, 191].

Concurrent execution

Dickerson et al. [53] introduced a technique based on transactional boosting methodology [75] that allows miners to execute non-conflicting smart contracts in parallel and then capture the parallel execution schedule for validators to reach deterministic results. This approach aims to improve throughput (the number of transactions per second) for block minders and transaction validators by executing irrelevant smart contracts concurrently. The paper recommends that miners execute smart contracts code as speculative actions to avoid shared data storage problems. If data conflicts are detected during the runtime, it can resolve by scheduling and delaying or rolling-back. Miners record the locking schedule of parallel execution in the corresponding block. It creates a happens-before graph of transactions for validators. Validators transform this schedule into

a deterministic parallel for-join program to validate the block concurrently. They made various benchmarks with smart contracts samples with different transactions and degrees of conflict to evaluate their approach. Testing based on three concurrent threads shows that the mining and validation processes can be accelerated up to 1.33x and 1.69x.

To introduce the three smart contracts execution model, Yu et al. [191] suggested breaking down the system states into three hierarchical states: account state, local state, and federal state, where the account state is the smallest state unit. Any changes in account state lead to the change in the local state, and any change in the local state ends with the change in the federal state. It improves performance by separating the execution of smart contracts from state management and offers a pipeline model to validate and create blocks in parallel.

1. Sequential execution of smart contracts: This is based on smart contracts execution in Ethereum. It is beneficial for the smart contracts that their execution time is quick, as this model has the longest time for block validation.

2. Parallel execution of smart contracts: If a transaction is recognized as a smart contract, all the nodes execute the smart contract simultaneously so that the local state will update concurrently. Nodes verify the local condition by comparing it with the block state.

3. Non-blocking execution of smart contracts: By separating execution of smart contracts from building blocks processes, expedite the block building and validating process. In this model, nodes do not wait for the current smart contract to finish the execution, and they accept the next transactions immediately. Besides, this study suggests decoupling block building from state maintenance in blockchain design. It introduces a new blockchain called SBC (State Blockchain) to handle state maintenance, including state synchronization and storage.

Anjana et al. [14] proposed using Software Transactional Memory (STM) [160] systems to execute smart contracts' non-conflicting transactions concurrently. There are two kinds of miners in the presented system: serial miner and concurrent miner. Both types of miners have access to the equivalent set of transactions. Serial miner executes transactions serially, whereas the concurrent miner executes the transactions concurrently. The concurrent miner must only choose non-conflicting transactions for executing concurrently. If two transactions access the shared data object and at least one of them performs the write operation on it, they would conflict with each other. The concurrent miners recognize conflict transactions with the help of an optimistic STM system. The validation of transactions can also be executed concurrently. There is a chance that validators re-execute suggested transactions from miners with different orders and get the different results that lead to the block rejection. To avoid this, the authors suggest concurrent miners provide a conflict graph. The conflict graph comprises all the dependencies between conflict transactions in the form of an adjacency list. Briefly, every concurrent miner presents a conflict graph and the set of proposed transactions, the hash of the previous block, and each shared object's final state.

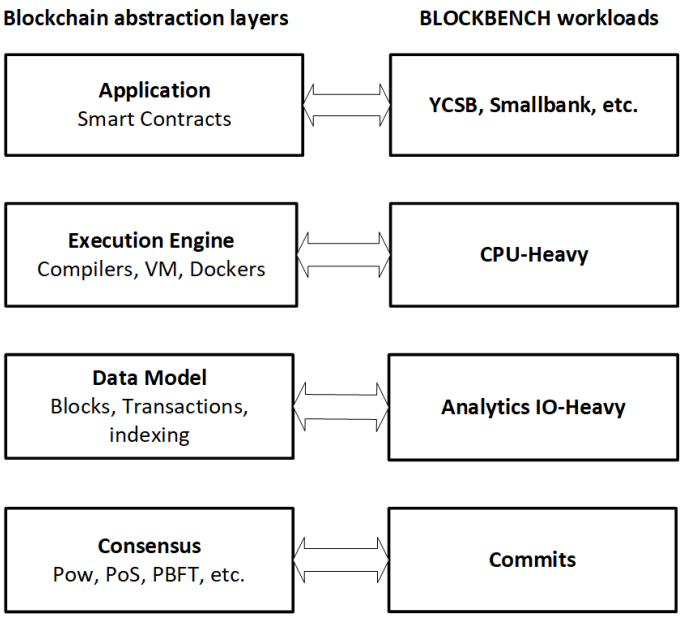

Figure 2.10: BLOCKBENCH abstractions layers and correlated workloads [55].

Performance analysing frameworks

BLOCKBENCH: BLOCKBENCH [55] is a framework for performance analysis of private blockchains. Any private blockchain can be integrated into BLOCKBENCH; however, the paper evaluated three blockchain platforms: Ethereum Geth (Go-version), Hyperledger Fabric, and Ethereum Parity. The study presents four main abstraction layers for blockchain and then accordingly designed four workloads for blockchain. Figure 2.10 explains blockchain abstractions layers and correlated workloads. To evaluate represented platforms, they implemented the smart contracts related to each workload in two versions of Solidity for Ethereum and Parity and Golang for Hyperledger. Table 2.7 shows the implemented smart contracts in BLOCKBENCH.

BLOCKBENCH measured throughput, latency, and scalability for analyzing the performance of private blockchains as following:

- Throughput: The number of successful transactions per second.
- Latency: Response time for each transaction.
- Scalability: Measuring the changes in throughput and latency when the number of nodes and the number of concurrent loads raises.

The evaluation results indicate that Hyperledger Fabric performed better in terms of performance metrics compared with Ethereum Geth (Go-version) and Ethereum Parity. Still, Hyperledger Fabric failed to scale up to more than 16 peers.

Real time monitoring: Zheng et al. [202] discuss why Gas is not a suitable metric because it is biased and inaccurate. They present two groups for the blockchain systems' performance metrics: overall performance

Table 2.7: BLOCKBENCH implemented smart contracts [55].

Smart Contract		Description
Macro benchmark workload	YCSB driver [45]	Key-value store
	Smallbank [30]	OLTP workload
	EtherId	Name registrar contract
	Doubler	Ponzi (pyramid) scheme
	WavesPresale	Crowd sale
Micro benchmark Workload	VersionKVStore	Keep state's versions
	IOHeavy	Read and write a lot of data
	CPUHeavy	Sort a large array
	DoNothing	Simple contract, which do nothing

for the users and detailed performance for the developers. Table 2.8. presents the subcategories of each group.

Table 2.8: Performance metrics based on [203].

Overall Performance Metrics	Detailed Performance Metrics
Transaction Per Second	Peer Discovery Rate
Average Response Delay	RPC Response Rate
Transaction per CPU	Transaction Propagating Rate
Transaction per Memory Second	Contract Execution Time
Transaction per Disk I/O	State Updating Time
Transaction per Network Data	Consensus-cost Time

They also offer a real-time performance monitoring framework to monitor blockchain systems' performance, which is scalable and carries a lower overhead. Here is a concise explanation of every part of the framework.

- Validating Peer is the primary underlying part of the blockchain to verify and execute transactions and smart contracts. The framework concentrates on its performance to calculate the performance of the entire blockchain.

- Log Parser/Analyzer is a terminal corresponding to each validating peer. This part collects the log data regarding the validating peer and hardware consumption.

- Synchronize Peer helps collect the rest of the data that validating peer did not catch without making extra overhead.
- Data Collector/Calculator collects data from a log analyzer and synchronizes peers and determines the performance metrics.
- Web Frontier is the user interface to visualize the collected performance parameters.

Finally, they monitored the performance of 1000 smart contracts on four different blockchain platforms, Ethereum, Parity, Hyperledger Fabric and CITA, based on mentioned performance metrics and the proposed framework metrics. Table 2.9 shows overall performance of all platforms.

Table 2.9: Performance of different platforms based on real-time monitoring [202].

Blockchain	Tps	TPC	TPMS	TPDIO	TPND
Ethereum	5.55578	0.00195	0.01060	0.26573	0.22206
Parity-pow	3.95500	0.00140	0.06814	0.00264	0.07167
Fabric	600.611	2.65340	4.28265	0.13816	0.10122
CITA	256.636	1.33393	0.43244	0.59888	0.16208

Besides, other studies have presented a tool or framework to measure and analyze the performance of smart contracts [55, 202]. Blockbench [55] is a framework for measuring the performance of private blockchains. They state that any private blockchain can be integrated into BLOCKBENCH. They evaluated three blockchain platforms: Ethereum Geth (Go-version), Hyperledger Fabric, and Parity.

Zheng et al. [202] presented a real-time performance monitoring framework to monitor the performance of blockchain systems. The implemented tool is scalable as we can easily extend it to monitor new peers or blockchain systems. Also, the framework has an insignificant performance impact on the running blockchain systems, and thus it has low overhead, and it is suitable for real-time monitoring.

2.2.5 Decentralized applications based on smart contracts

The evolution of smart contracts has provided the required infrastructure for non-monetary blockchain-based applications in various domains. Blockchain desirable features, such as distributed operation, immutable audit trail, data provenance, security, and privacy, have made it a suitable alternative for traditional centralized applications, and the evolution of smart contracts has made it possible. This section reviews the main application domains that use blockchain and smart contracts. They are categorized in different groups, such as IoT presented in table 2.10, healthcare presented in table 2.11, Supply Chain presented in table 2.12, Business process management presented in table 2.13, record-keeping presented in table 2.14, voting presented in table 2.15, and identity management presented in table 2.16.

Table 2.10: IoT decentralized application.

Reference	Application Feature
Ouaddah et al. [126]	FairAccess is a framework for access control and authorization mechanism based on ethereum smart contracts for IoT systems.
Ouaddah et al. [127]	Pseudonymous and privacy-preserving authorization and access control management framework for IoT systems.
Xu et al. [185]	BlendCAC, federated capability-bases Access control mechanism for IoT systems based on smart contracts.
Cha et al. [36]	Privacy preserved connected gateway for IoT devices. Using smart contract for managing the IoT device data and defining privacy policies.
Huh et al. [82]	Using Ethereum smart contract to store the data produces by IoT device, and then configure and define the behavior of the IoT devices.
Ruta et al. [147]	A service-oriented architecture based on semantic blockchain and implementing smart contracts for registration, resource discovery, resource selection, and payment.
Lombardi et al. [100]	Blockchain-based infrastructure to create a reliable and low-cost transactive energy. Energy trade and energy auction handle by smart contracts.

Table 2.11: Healthcare decentralized application.

Reference	Application feature
Azaria et al. [15]	Patient-centric medical data access control and sharing system (Medrec) based on Ethereum platform. It presents a method to provide anonymous data as an award for miners.
Xia et al [183]	Big Medical data sharing, provenance and audit trail on a cloud platform (Medshare).
Xia et al. [184]	Using permissioned blockchain and integrating identity management service to access to the shared EHR data.
Rouhani et al. [142]	DAC Hyperledger fabric permissioned blockchain (MediChainTM).
Mikula and Jacobsen [113]	A prototype based on Hyperledger fabric and ChainCodes, which represents identity and access management in healthcare domain.
Yue et al. [194]	Patient-centric mobile application to share medical data based on Indicator-Centric Schema.
Peterson et al. [132]	Securely sharing medical data exchange and presenting Proof of Interoperability consensus method by considering FHIR [55]
Theodouli et al. [170]	Proposing a blockchain-based architecture design for sharing medical data.
Nugent et al. [123]	Using smart contract to solve data manipulation issue in clinical trials and proving a trusted immutable record of data.
Shae and Tsai [157]	Presenting a blockchain platform and architecture for clinical trial and precision medicine. Include four main components: 1) blockchain-based parallel computing, 2) blockchain application data management component, 3) identity management component, 4) trusted data sharing management component.
Zhang et al. [197]	FHIRChain,a blockchain-based architecture design based on ONC (Office of the National Coordinator for Health Information Technology) requirements and FHIR [20] standard.
Zhang et al. [198]	Introducing evaluation metrics to analyze Dapps (Decentralized blockchain-based application) in healthcare domain.

[18]https://www.hl7.org/fhir/overview.html
[19]https://www.hl7.org/fhir/overview.html
[20]https://www.hl7.org/fhir/overview.html

Table 2.12: Supply chain decentralized application.

Reference	Application feature
Chen et al. [37]	utilizing smart contract and blockchain for quality management in supply chain. The real-time quality management, process the quality, and the product quality are monitored by smart contracts and unqualified products automatically reject by the smart contract.
Bocek et al. [21]	Using smart contract and IoT sensors to assert data immutability temperature records in pharmaceutical supply-chain and reducing the cost.
Korpela et al., [91]	Investigates using smart contracts for documentation collection trade and automation of transactions, or in general, digital supply chain integration based on DBE framework [92].
Ruta et al. [146]	A semantic-enhanced blockchain platform that supports ontology-based object discovery.
Kim and Laskowski [88]	Analyze a traceability ontology and translate it to a smart contract to achieve supply chain provenance trace and enforce traceability constraints using Ethereum platform.
Figorilli et al. [59]	Tracing electronic records from standing tree to the whole chain of steps using RFID sensors and Azure blockchain workbench and two main smart contracts, timber marking and cutting trees.

Table 2.13: BPM (Business Process Management) decentralized application.

Reference	Application feature
Weber et al. [178]	Converting the BPMN model to a factory contract and then implementing a factory contract and running its instances as a smart contract on the blockchain to address the trust issue in the collaborative business process.
García-Bañuelos et al. [62]	Presents an approach to convert a business process model into a Petri-net model and then compile it to a minimized solidity smart contract that encodes the preconditions for executing each task of the process using the space-optimized data structure.
Pouheidari et al. [135]	Investigating a case study of Execution of Untrusted Business Process on Hyperledger Composer [79] and Hyperledger Fabric [6] as a permissioned blockchain and investigate the advantages of using permissioned blockchain for the collaborative business process.
Mendlin et al. [111]	A survey paper that investigates challenges and opportunities of blockchain for BPM.
López-Pintado et al. [101]	A blockchain-based Business Process Management System. It uses solidity smart contracts to encode the state of the business process instance and execute workflow routing.
Ribma et al. [138]	Comparing the cost of computation and storage of business process execution on blockchain and cloud platforms. They represent a cost model based on the cost of deploying smart contracts and executing process coordination.

Table 2.14: Record keeping decentralized application.

Reference	Application feature
DÁngelo et al. [48]	Using blockchain-based logging system to log and record the interactions and employing smart contract as an arbitrator to verify the stored events, detecting (Service Level Agreements) SLA violations and calculating compensations.
Lemieux [98]	A theoretic framework to evaluate the potential of the blockchain and smart contract as a decentralized trusted record keeping system.
Lemieux [98]	Creating and updating records on blockchain using smart contracts by encoding procedures that run on a multi-stakeholder network.
Guo et al. [67]	Using blockchain as an event recording system for autonomous vehicles. A “proof of Event” mechanism with Dynamic Federation consensus records is suggested to resolve accidents disputes.

Table 2.15: Voting decentralized application.

Reference	Application feature
McCorry et al. [110]	Completely untrusted protocol based on smart contract for boardroom voting that guarantees voters privacy. The smart contracts handle the voting protocol, monitor the election process, create and verify the zero-knowledge proofs [19].
Shah [158]	A blockchain-based voting system that uses smart contract codes for verifying and adding a record of votes to the blockchain.
Chen et al. [40]	Using smart contract to solve the problem of the bid price leakage before the specified deadline. The smart contract includes the auctioneer's address, start time, deadline, address of the current winner and current highest offer.

Table 2.16: Digital identity decentralized application.

References	Application Feature
Yasin and Liu [190]	A framework for online identity that uses smart contracts to calculate personal online ratings based on the aggregated data from all the digital identities of the user.
Al-Bassam [5]	Using web-of-trust and smart contract to manage identities and their attributes that stored on the blockchain and create a decentralized PKI and identity system.
DeCusatis and Sager [50]	Investigating a prototype that implements an end-to-end user identity management system using Hyperledger Fabric. Suggesting identity-based network segmentation and traffic separation to increase the security and network tolerance against DDoS attack.
Grüner et al. [65]	First, it discusses the decision of using blockchain based digital identity system. Second, it analyzes uPort[103] Sovrin[137], and ShoCard[162]. The analysis result indicates uPort can be best fit with blockchain technology. Strict trust framework of Sovrin may not work with blockchain and decentalized nature.
Mühle et al. [115]	Discussing the main components, architecture and actors of a self-sovereign identity management system based on blockchain and smart contracts as well as authentication and storage solutions.

Although researchers considered blockchain and smart contracts for various applications, access control management, sharing data and resources, and tamper-proof record-keeping are the primary focus of decentralized applications, mainly in healthcare and IoT. So despite the variety in the applications, their focus is narrowed down to these three aspects.

2.3 Access control

Access control [150] is a security technique that controls access requests toward protected resources in a system. These resources could be physical or logical. Physical resources are like buildings, rooms and physical IT assets. Which can be protected using hardware equipment along with software controls. Logical resources are like data and computer networks. When a request comes to the system and asks for access to a resource, the access control component decides to accept or deny the request based on the implemented access control mechanism or the user or component who has sent the access request. There are two main parts involved in access control implementation. First, who has access to which resources, under what circumstances and what

types of access are allowed? This part is implemented as a policy, and it is called the access control policy. The second part is access control enforcement, which enforces the system only to accept access requests matched with access control policy [165].

This section presents answers to the following research questions.

- What are the problems with current access control systems?
- How blockchain can help to solve these problems?
- What are the challenges for implementing an access control system based on blockchain?
- What are the features of implemented prototype systems?

2.3.1 Access control models

Access control policies are commonly grouped into four models that are also referred to as access control models. System designers choose an access control model based on the resources' sensitivity and implementation cost.

- Discretionary Access Control (DAC): It controls access based on the requestor's identity and access rules. In DAC, access to a resource is at the discretion of the resource owner, granting access to another entity to access the resource. Anyone who creates the resource is the owner of the resource.
- Mandatory Access Control (MAC): Unless DAC, in MAC, the resource owner is not the one who creates the resource, the organization or the company are the resource owners and decide on whom to share resources. MAC is implemented by security labels and controls access based comparing them. These labels indicate the sensitivity level of the resources as well as some information regarding the category of the data. Access propagation is not possible in MAC.
- Role-Based Access Control (RBAC): RBAC controls access based on the users' role in the system or organization. RBAC is implemented by defining rules that define the access rights of each role.
- Attribute-Based Access Control (ABAC): ABAC controls access based on the attributes of the user, resource and environment. ABAC is implemented by specifying policies that define the access to the resources based on these attributes. ABAC is discussed with more details in chapter 6.

2.3.2 Implementing access control

Access Control Matrix (ACM) is a matrix [165] that stores each subject's rights with respect to every object in the system. Rows in the matrix are the users or subjects, and the columns correspond to the resources or objects. The entries in the matrix determine access rights to the corresponding subject and object. It is a large and sparse matrix, and many of the entries are null. There are two methods to represent ACM efficiently: Access Control List (ACL) and Capability-based [149].

ACL

ACL is an implementation of ACM that focuses on the columns or resources. For every resource or object, ACL stores a list of users who have access to the resource and their access rights on a file. In ACL system, it is the service provider's responsibility to check if the user is authorized.

Capability-based

Capability-based [70] or capability list is an implementation of ACM that focuses on the rows or objects. A capability can be considered a pair (x, r) where x is the name of a subject, and r is a set of subjects' rights. Capability-based is implemented by issuing unique access tokens to the subjects with capabilities. The access token references the subject along with an associated set of access rights. In the implementation of Capability-based access control, anyone who presents the capability can access the resource. It is not important who presents it. Therefore capabilities must be unforgeable tokens to avoid invalid access permissions. This implementation eliminates the need for authentication and makes it a potential solution for access control for dynamic systems such as IoT, in which a centralized authorization server can cause performance issues. Unlike ACL, that service provider has to check the user's authorization; in capability-based systems, the user must present its capability (access token) to be authorized. Due to flexibility and more manageable features in capability-based access control, it is widely used in the blockchain field for IoT applications [99, 118, 185].

2.4 Data trust based on blockchain

2.4.1 Data trust definition and architecture

Trust is a multidisciplinary and multifaceted concept that has been defined in various disciplines, such as sociology, economics, psychology, computation, information and computer science, to model different types of relationships. Trust's definition can be challenging since it embraces the facets of ethics, emotions, values, and it combines various disciplines. Fundamentally, trust is a relationship between trustor and trustee in which the trustor relies on the trustee based on a given criterion. Cho et al. [42] summarizes the trust definition after studying trust across multiple disciplines, as: "Trust is the willingness of the trustor (evaluator) to take a risk-based on a subjective belief that a trustee (evaluatee) will exhibit reliable behaviour to maximize the trustor's interest under uncertainty (e.g., ambiguity due to conflicting evidence and/or ignorance caused by complete lack of evidence) of a given situation based on the cognitive assessment of past experience with the trustee". Typically, digital trust is considered a computational value established from a relationship between trustor and trustee, and measured by trust parameters and evaluated by a defined mechanism [188].

Since the development of big data and data science, many technologies have advanced rapidly. However, the process of collaborative data sharing, reusing data, and data privacy protection is still facing serious problems. Data trust is a mechanism to address these problems by providing a structured and solid framework

for data stewardship and facilitating the process of access to data.

The basic idea is to have a well structured, transparent and trustable framework for data stewardship. Sharing data might not only compromise the interests of data subjects at risk of exposing their personal information, but it could also lead to the loss and fine for the organization and consequently reputational damage. We expect to mitigate some of the perceived risks of data sharing with data trust. Moreover, the framework must be designed to respond to data owners' concerns and data subjects by providing ethical principles underlined in the data trust framework and ensuring the data users regarding the trustworthiness and quality of the data. Data owner refers to an individual or an organization that owns and controls the data. Data subjects are individuals that the data is related directly or indirectly to their personal information. Data users or data consumers are individuals or organizations that consume the data for data scientists and analyzing purposes.

Although data trust could be a law arrangement or contractual agreement, it is possible to be programmed into the architecture that satisfies specified requirements. It is essential to understand the interaction between different actors and components and establish mutual trust. O'Hara proposes eight essential properties that underlie data trust architecture [124], (1) discovery, (2) provenance, (3) access controls, (4) access, (5) identity management, (6) auditing of use, (7) accountability, (8) impact. He proposed the Web Observatory as a candidate technology to carry out a data trust operation.

- Discovery refers to the process of discovering the quality and properties of data by data users in the first place
- Provenance refers to the ability of data users to access the historical record and metadata about the data
- Access control refers to the ability of data owners to control and manage access permissions toward their data
- Access refers to the mechanism that provides access for data users
- Identity management refers to the ability of data owners to identify and authenticate data users
- Auditing of use refers to providing a transparent history of data usage
- Accountability refers to achieving accountability by access control and auditing of use
- Impact refers to assessing the value, usage and misuse of data through recorded information in the data trust

Figure 2.11 illustrates O'Hara's architecture for data trust called Data Trust Portal (DTP) for sketching out the above features inspired by web observatory infrastructure. The data does not store in DTP, data owners hold the data, and they are responsible for their data protection and the implemented interface method

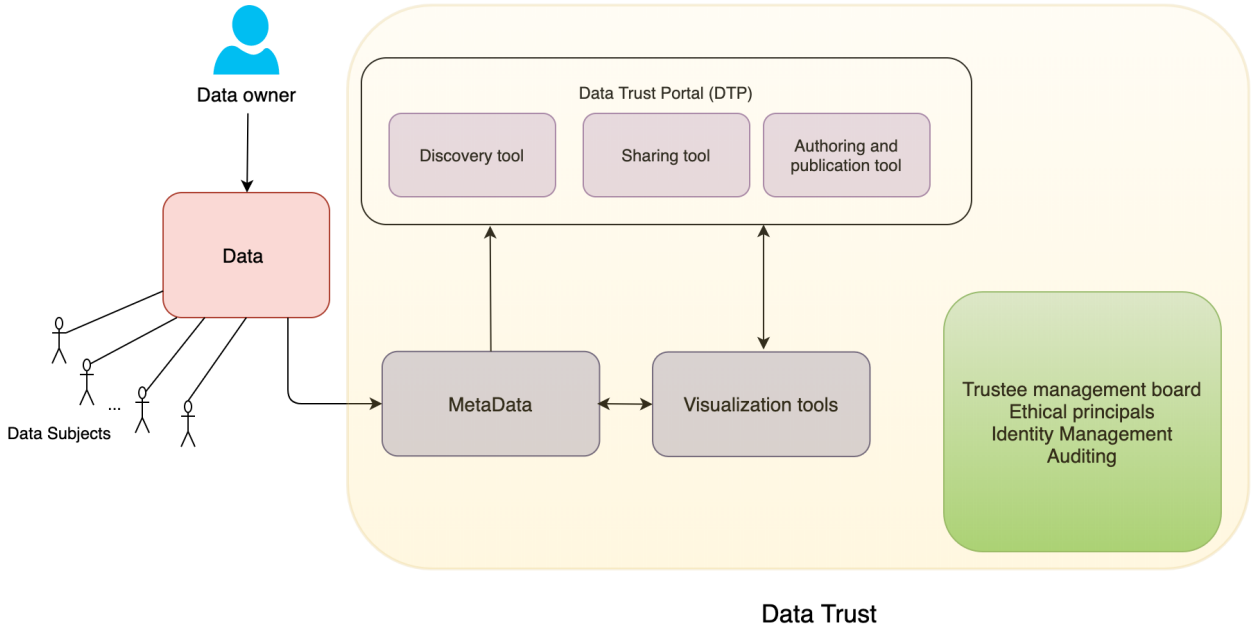

Figure 2.11: Data Trust Portal (DTP) architecture by O'Hara [124].

to provide access. DTP is a platform for implementing secure discovery and sharing protocol using metadata about data properties and their provenance. Alsaad et al. [9] also proposed institutional repositories, which is a mature platform and open framework, as an alternative technology for data trust infrastructure.

Stalla-Bourdillon et al. [164] introduces a workflow that addresses data protection by design approach to achieve efficient data usage, sharing, and reusing. The study emphasizes a design with well-defined data governance roles and processes. They represent data trust through three core layers as expressed in figure 2.12 : (1) the data layer (2) the access layer (3) the process layer.

In the data layer, interested parties sketch out the steps of creating data pools by making sure that everyone is aware of the legal obligations for data protection by design, specifying the authorized individuals to decide and act on the pooled data, and preparing data for sharing by employing technical procedures to remove personal identifiers, such as de-identification and anonymization [119]. In the access layer, the data become discoverable for eligible parties by specifying standardized access through centralized or peer to peer technical solutions complemented by monitoring and auditing processes. In the process layer, data trust approves the pooled data for re-usage. This layer is responsible for controlling data usage through standardized risk assessments and ensuring that data are tailored to queries.

2.4.2 Blockchain as a infrastructure for data trust

We can utilize blockchain as a data trust interface between data controllers, data subjects and data users. The distributed, secure and reliable nature of the blockchain can reinforce the data trust framework's trustworthiness.

As I mentioned in the previous section, O'Hara [124] introduces eight properties that system architectures

should consider for data trust architecture, including (1) discovery, (2) provenance, (3) access controls, (4) access, (5) identity management, (6) auditing of use, (7) accountability, (8) impact. Some of these properties, such as provenance, auditing of use, and accountability, already exist in the blockchain since blockchain provides a secure, tamper-proof, immutable record of transactions. All blocks are linked together through their hash values. Some other properties, such as discovery, access control, access, and impact, could be implemented through smart contracts and executed on permissioned blockchain. Identity management can be implemented through membership service in permissioned blockchains. Ultimately, accountability can reach because multiple peers verify transactions through consensus mechanisms, and the ledger is maintained precisely through cryptographic methods. Moreover, every peer has a copy of the ledger, and the network can easily recognize any inconsistency.

2.5 Decentralized access control systems

Access control systems have been created to protect system resources from unauthorized access. Studies [47, 57, 107, 175] have shown that current access control systems face numerous problems, such as difficulty in interoperability between multiple organizations, lack of privacy, inefficiency and trust distribution, and inefficiency. The majority of these problems stem from the centralized nature of current access control systems and third parties' presence.

Blockchain salient features, including decentralization, immutability, durability, security, and distributed ledger, show great potential to address these problems.

In this section, the studies that present a blockchain-based solution for access control systems are reviewed. The access control systems follow one of the available access control models based on the system policies, security procedures within the organization, and the level of the resources' sensitivity. These studies are categorized based on the applied access control model, the domain of the application.

Many studies on blockchain technology have focused on presenting an access control system either in the context of specific applications, such as healthcare [15, 47, 136, 142, 183, 199], IoT [54, 56, 105, 122, 126, 133, 200], and cloud federation [6, 58] or they have introduced a general access control system, which can

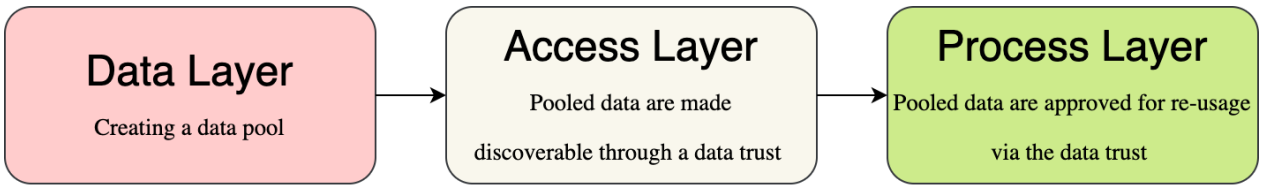

Figure 2.12: The workflow of data protection by design approach [164].

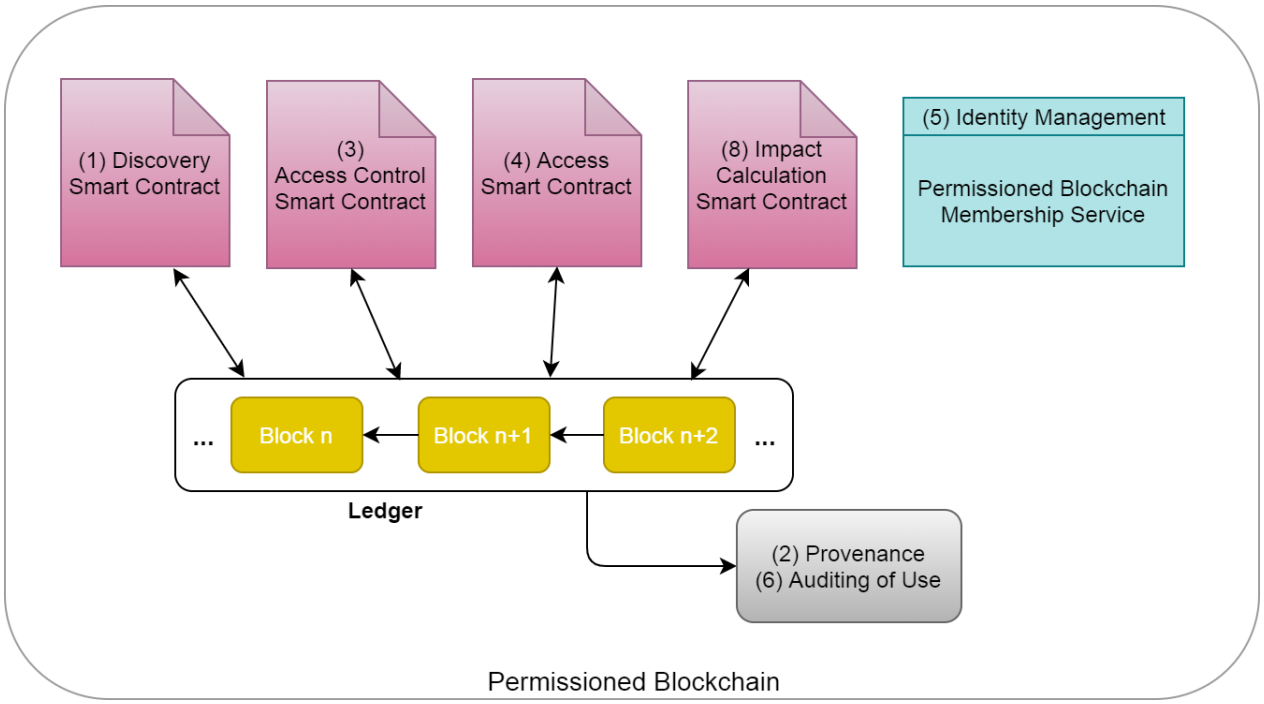

Figure 2.13: Blockchain as a infrastructure for data trust.

be employed for different applications. In my previous study [144], the state of the art of access control systems based on blockchain has been investigated. This section overviews similar studies, which present an attribute-based access control based on blockchain. As illustrated in table 2.17, many studies use the attributed-based method for their access control system because of the granularity, flexibility, and dynamic features that ABAC provides.

Guo et al. [66] proposed a hybrid architecture for access control over Electronic Health Record (EHR) data using blockchain and edge nodes. The blockchain acts as a tamper-resistance verification component to verify identities and access control policies. The edge nodes record the EHR data off-chain and enforce the access control policies. The smart contracts include the address of EHR data on the edge nodes by using one-time self-destructing URLs [21]. Based on experiments of performance results against unauthorized retrieval for the average transaction processing time was 40 ms, and the average response time was 30 ms. Also, to test their system's scalability, they test the system with five patients to 320 patients. The result did not influence the response time, indicating the scalability of the system.

Zyskind et al. [210] considered the blockchain technology as an access control moderator, complemented by off-chain storage. Blockchain clients represent users who contribute their data to a service provider and are the owners of their data. Based on that assumption, this solution is intended to empower users, giving them information about which data is collected by third parties and how their data is used. To reach this goal, each data owner can issue transactions used to modify the set of permissions granted to a service or

[21] https://1ty.me/

entity. All transactions are stored on the blockchain, allowing for auditability and traceability.

Zhang et al. [200] proposed a solution focused on IoT blockchain-based access control [200]. The authors introduce the concepts of *Judge Contract* (JC), *Register Contract* (RC) and *Access Control Contracts* (ACC). Access control contracts record access control policies for a subject-object pair. In their system, both JC and RC are essential pieces regarding achieving a distributed and secure access control. The JC receives misbehaviour reports and applies penalties on them. The RC stores the misbehaviour information from the JC and handles it through the judging method. Furthermore, it stores information such as name, subject, object, and smart contract for access control.

Zhu et al. [206] presented a Transaction Based Access Control (TBAC) system, which integrates the ABAC model into the bitcoin blockchain. There are four implemented transactions, subject registration, object escrowing and publication, access request and grant. They also introduce a cryptosystem associated with their system as a supplementary security layer. They evaluated the system in terms of security; still, they did not examine the system's performance and scalability.

In the federated cloud services, access control enforcement is still vulnerable to privacy breaches. Alansari et al. [6] introduced an attribute-based access control system based on the Pedersen commitment scheme [112] and blockchain. They designed the system to keep the users' attributes private from the federated organization. Users' identity attributes and access control policies are stored on the blockchain to assure their integrity. They also hired Trusted hardware technology to guarantee the integrity of the policy enforcement process.

Zhang et al. [198] presented an architecture for granular access authorization supporting flexible queries to implement secure authorization at different levels of granularity. The designed architecture offers a capable infrastructure without needing the public key infrastructure (PKI), as a result decreasing the computation time; therefore, it is suitable for the devices with limited resources in EMR systems. Their system can efficiently respond to a requester without exposing unauthorized private data.

Maesa et al. [107] proposed an access control service on top of Ethereum [22]. They utilized blockchain to deploy smart contracts representing access control policies presented in eXtensible Access Control Markup Language (XACML) [12] performing the access decision making. Such smart contracts are called *Smart Policys (SPs)*. Thus, SPs are responsible for the policy evaluation, embedding a Policy Decision Point (PDP) for a particular access control policy. Every time the system evaluates an access request needs to make an access decision, and the blockchain executes it in a distributed way. The decision is made based on information concerning the users. For this purpose, they introduce the concept of Attribute Managers (AMs) as the components that manage the attributes of the entities involved in the process, such as subjects, resources, and environmental context. AMs updates and retrieves their values created by an entity, the Attribute Provider (AP). Implementation based on the Ethereum blockchain is costly because, for every operation, the clients must pay a fee in terms of gas. Although this cost is generally unavoidable for public

[22] www.ethereum.org

blockchain systems, for implementation based on permissioned implementation, this unnecessary additional cost is imposed on the system.

The privacy and security of the data created by IoT devices are the primary concerns of the IoT system due to the extensive scale and distributed nature of IoT networks. Many studies have considered blockchain to provide secure access control to the IoT data to protect users' privacy. Dukkipati et al. [56] provided a solution to store system policies off-chain while Pinni et al. [133] stores the policies on the Ethereum platform, which is less vulnerable to security breaches, although more costly. Ding et al. [54] focused on simplifying the access control protocol to make it lightweight and fitting for IoT devices with limited computing and energy resources.

2.5.1 Challenges in implementing access control system based on blockchain

In this section, I discuss the open challenges associated with blockchain-based solutions [144] for access control systems. In the following, I discuss a list of the existing challenges.

- Off-chain and on-chain integration: Blockchain is not a suitable infrastructure for storing a large volume of data. The original data needs to keep in secure off-chain storage, access policies, a hash of the data, and references to the data record on blockchain [142]. The safe integration between on-chain and off-chain storage is challenging, and it needs further investigations. Using trusted hardware technology such as intel SGX can be admitted to keep the integrity of the system.

- Blockchain vulnerability: Besides all the desirable advantages of blockchain with empowering smart contracts, it is still challenging to implement non-vulnerable smart contracts [104] in the blockchain. Subsequently, designing methods and tools to improve smart contracts and blockchain security is highly investigated [10, 20, 23]. Especially, the importance of accuracy of smart contracts significantly raises when it comes to implementing an access control system.

- Transaction transparency: One of the main reasons blockchain has become successful was providing transactional transparency; however, this is not favourable from the enterprise perspective and privacy point of view. This is why we have witnessed the advent of permissioned platforms, which support transaction privacy and private data. However, it requires sacrificing pure decentralization and accepting hybrid centralized and decentralized solutions.

- Blockchain performance: The performance of execution and validation of transactions has been raised recently by proposing lighter consensus methods and more efficient transaction processing flow in blockchain platforms such as Hyperledger Fabric. Despite recent studies in improving the performance of blockchain [14, 53], Still, the performance of the blockchain-based solutions cannot compete with the current centralized solutions. As we can see, most of the studies compare their given system's performance with other blockchain-based platforms, not existing decentralized solutions. To resolve the

performance and scalability problem [105] proposes an architecture based on multiple blockchains on the cloud systems. The designed architecture comprises various layers, including the device layer, edge networking layer, fog layer, core network layer, and cloud layer. Multiple blockchains are implemented in the fog layer based on different edge networks to keep access control and key management transactions. The cloud layer includes all the blockchains in the fog layer and implements the interactions among these blockchains. Simulation results show that dividing a load of block mining into multiple blockchains decreases the transaction collection time, blocks mining time consumption, and enhances the system's performance.

- Access control methods: As in traditional systems, most trending access control systems use ABAC as an access control method. Based on table 2.17, it is clear that ABAC is also very common for implementing an access control system using blockchain; however, current access control techniques which are static might be inadequate for future designs [31] and more dynamic access control methods, one in which resources define their access control, might be required. Integrating blockchain with dynamic access control approaches could be an exciting area to investigate in the future.

2.6 Contribution of the work presented in this chapter

This chapter comprehensively and systematically reviewed smart contracts topics from different angles. The implementation aspects, security enhancement approaches, performance improvement studies and decentralized application based on smart contracts were discussed thoroughly. I studied the design, structure, features, and implementation of smart contracts.

This chapter also investigated centralized and traditional access control systems' problems and discussed how we could address them with blockchain technology. A review of access control studies based on blockchain by presenting state of the art and future direction was presented.

The contribution of this chapter has been published in two review papers as below.

Rouhani, Sara, and Ralph Deters. "Security, performance, and applications of smart contracts: A systematic survey." IEEE Access 7 (2019): 50759-50779, doi: 10.1109/ACCESS.2019.2911031.

Rouhani, Sara, and Ralph Deters. "Blockchain based access control systems: State of the art and challenges." In IEEE/WIC/ACM International Conference on Web Intelligence, pp. 423-428, 2019, doi: 10.1145/3350546.3352561.

Table 2.17: A summary of blockchain-based access control applications.

Research paper	Domain	Access control method	Privacy Support	Scalability
Maesa *et al.* [106]	General access control	ABAC	x	x
Guo *et al.* [66]	General access control	ABAC	x	✓
Laurent *et al.* [109]	General access control	Access Control List	✓	x
Maesa *et al.* [107]	General access control	ABAC	x	✓
Guo *et al.* [68]	General access control	ABAC	✓	x
Lee *et al.* [97]	General access control	Role-based	✓	x
Jemel and Serhrouchni [84]	Data sharing	ABAC + Attribute-based Encryption	✓	x
Wang *et al.* [175]	Data sharing	ABAC + Attribute-based Encryption	✓	x
Zhu *et al.* [206]	Resource sharing	ABAC	✓	x
Hu *et al.* [78]	Knowledge sharing	Fine-grained	x	x
Ferdous *et al.* [58]	Cloud federation	-	x	x
Alansari *et al.* [6]	Cloud federation	ABAC	✓	x
Zhang and Posland [198]	Health care	ABAC	✓	x
Rouhani *et al.* [142]	Health care	Role-based	✓	✓
Asaph *et al.* [15]	Health care	-	✓	✓
Xia *et al.* [183]	Health care	-	✓	x
Dagher *et al.* [47]	Health care	Role-based	✓	✓
Rajput *et al.* [136]	Health care	Role-based	✓	✓
Zyskind *et al.* [210]	Mobile applications	Policy-based	✓	✓
Novo [122]	IoT	-	x	✓
Outchakoucht *et al.* [128]	IoT	Policy-based	x	x
Zhang *et al.* [200]	IoT	Policy-based and dynamic access control	x	x
Xu *et al.* [185]	IoT	Capability-based	✓	✓
Nakamura *et al.* [118]	IoT	Capability-based	x	✓
Dukkipati *et al.* [56]	IoT	ABAC	✓	x
Pinno *et al.* [133]	IoT	ABAC	✓	✓
Ouaddah *et al.* [126]	IoT	-	✓	✓
Ding *et al.* [54]	IoT	ABAC	x	✓
Ma *et al.* [105]	IoT	Generic	✓	✓
Rouhani *et al.* [145]	Physical access control	Role-based	x	✓
Es-Samaali *et al.* [57]	Big data management	ABAC	✓	✓
Xu *et al.* [186]	Space situation awareness	Capability-Based	x	✓
Paillisse *et al.* [129]	Multi-administrative domain	-	✓	x

3 Trust establishment in business process management

The progress in digitization and internationalization of business processes in various business-related areas, such as e-commerce and supply chain, led to increasing inter-organizational collaboration. In such a partnership, multiple organizations work together toward a common objective. Each organization carries out its own sub-processes, which are beyond the domain of influence of other collaborators. However, the result of these sub-processes affects the main collaboration's outcome. Thus still collaborators want to engage in the process, and they need trust [116].

Numerous studies examined the effects of the business processes integration on either firms' operations or business performance. As any level of integration can increase business performance and dimension. However, two prominent issues exist in almost every proposed integration approach, lack of shared repository and lack of trust. The first issue is the lack of a reliable, trustable, and tamper-resistant database. Despite advances in databases in terms of communication methods and encryption techniques, most business parties still store their data and transactions in their separate confidential databases.

The main problem with such an approach is that it is complicated to track the business's state and trace transactions between collaborative parties. Another problem is that cooperative parties may tamper with stored data on their databases to cheat others and gain more profits. The second object is the lack of trust between collaborative business parties. We are considering a collaboration between several organizations or individuals while some might be competitors. Typically, there are many conflicts to select an orchestrator. In such an untrusted network, who should be held as a central controller core and get everyone's agreement?

A collaborative business process comprises multiple competitors. Each party must ensure that the data related to their business contracts, assets, and transactions are kept private from their rivals. Hence, to implement the collaborative business process on the permissioned blockchain, it is necessary to provide reliable access control administration.

Lack of trust and scattered data on different isolated databases is a real issue in collaborative business processes. Blockchain technology offers features to establish trust between multiple cooperators in sharing collaborative business processes by monitoring the process of using and sharing data. Blockchain applies consensus mechanisms to eliminate the trusted third party's necessity to reach agreements and confirm transactions. It provides a distributed shared ledger, which facilitates the task of the process monitoring for the parties. The smart contract can be utilized to define the guaranteed autonomous programs. Besides, using a permissioned blockchain for managing the access control can keep the privacy of the data.

Blockchain's inherent features can potentially contribute to feasible solutions for these obstacles in col-

laborative business processes.

Blockchain platforms employ consensus mechanisms to validate the transactions in the untrusted network. Hence, they do not require trusted third parties anymore. Several parties can join the blockchain network, including a circle of business partners, without any concern about the potential of tampering or fraud. Because every party has a duplicate copy of the ledger, they can trace the process's state and all confirmed transactions reliably. We can deploy smart contracts to monitor and trace the business process flow. Furthermore, we can utilize smart contracts to implement complex business and regulatory rules.

In this chapter, I demonstrate how blockchain can impact untrusted business process management. I investigate the execution of a real-world business process on a permissioned blockchain. I examine the usage of a permissioned blockchain for access management and monitoring components to maintain the collaborative business processes. The examination is done by implementing the process of Order Processing on the Hyperledger Fabric Fabric as permissioned blockchain and Hyperledger Composer as the development framework are utilized.

In permissioned blockchain networks, only authenticated users can join the network. However, this is not sufficient for covering all business processes scenarios because every authenticated participant can still see all the data and transactions within the network. At the same time, we need to restrict access to the users who authenticated and joined the network. We need more granular access control in business process scenarios offered by permissioned blockchains.

3.1 Business process

A business process is "a collection of activities that takes one or more inputs and creates a valuable output to the customer" [71]. In other words, a collection of tasks and activities, relationship between different activities, specific inputs and outputs, and customer(s) define a business process. A collaborative business process is when multiple organizations work together toward a particular goal.

A business process regularly is modelled as a flowchart. Business Process Modelling and Notation Business Process Modelling and Notation (BPMN) [180] is the most common standard that present graphical illustrations of business processes. A BPMN process includes diverse types of elements: objects, sequence flows and message flows. An object itself can be an activity, an event, or a gateway.

3.2 Order processing model

Figure 3.1 shows the BPMN flowchart of the order Processing model. C, M, and L letters are assigned to the business model's activities, start event and XOR decision gateway to define which party can access those elements. These letters stand for the customer, manufacturer and logistics, respectively.

The customer can create an order to buy a commodity from the manufacturer. Then the manufacturer receives the order and assesses it. It can either reject or accept the order based on order assessment. If the

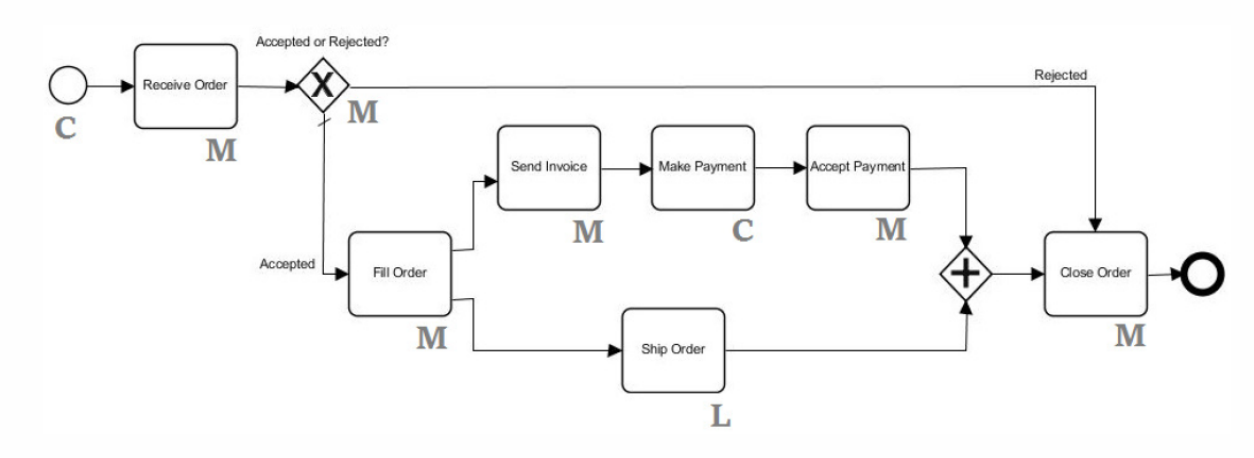

Figure 3.1: Order processing model.

manufacturer accepts the customer's request, it fills the order and sends the invoice back to the customer. The customer should then pay the fees, and the manufacturer should review and accept the customer's payment. After filling the manufacturer's order, the logistics ship the ordered commodity from the manufacturer to the customer. It is a parallel flow, so the Ship Order task can occur any time after the Fill Order task, independent from Send Invoice, Make Payment and Accept Payment tasks. Finally, if the manufacturer declines the order or when both the Ship Order and Accept Payment tasks are fulfilled, the manufacturer can close the order.

3.3 Privacy and access control in collaborative businesses

Privacy is an essential part of any practical collaborative business. Therefore, it is required to define access procedures for the deployed smart contracts as well. Consequently, we should define access policies to regulate access permissions on system resources, transactions, or smart contracts programs in the blockchain network. A few permissioned blockchains, such as Hyperledger Fabric, provide a structured infrastructure to manage access to different elements in blockchain networks. For example, authenticated users might be restricted from accessing the smart contract programs or private data stored on the ledger.

Although using a universal centralized access control system simplifies the administration task, some weaknesses are associated with this method, such as single point of failure and unreliability. Furthermore, in an untrusted collaborative process, allocating the access control authority to one contributor could cause unclarity and this lack of transparency may develop further issues.

In this study, I used Hyperledger Composer framework to determine different business network elements, including access control rules. Hyperledger composer is a framework for creating business applications on top of Hyperledger Fabric permissioned blockchain. Hyperledger Composer contains an Access Control Language (ACL) that provides a declarative definition of access control over all resources defined in the model file.

Furthermore, the advantages of using a decentralized access control system do not limit to collaborative processes. Therefore, we can extend its usage to specify different levels of access in intra-organizational scenarios.

3.4 Business network implementation

3.4.1 Defining the model as a business network archive

Within the Hyperledger Composer, each business model, which in this case study is an order processing model, is defined as a Business Network Archive (BNA) comprises a model file, a script file, an access control file, and query implementations. I defined the business model's distinguished aspects and stored them through these files, and then I compressed them as the BNA. Lastly, I deployed the BNA file on the Hyperledger Fabric blockchain platform.

I defined the business domain in the model file employing the Hyperledger Composer's object-oriented modelling language. The model file includes the definition of assets, participants, and transactions. For any collaborative business model, we need to define these resources in such a way that the business rules are followed accurately.

I considered the business flow as an asset, including: 1) Boolean properties for each business model factor, which requires business parties' actions, such as tasks and decision gateways. 2) Relationships to corresponding participants. 3) And required properties for describing the content of the business. Therefore, we can easily monitor and trace the business flow and define who can access assets' status with details. Table 3.1 represents the considered asset components for implementing the order processing model.

Status is an enumerated type that can be either active or closed. Once the customer opens an order, the order status is active, and when the order is closed by the manufacturer, the order status becomes closed. ID, name and description properties define an order. Although they are completely arbitrary, they are general attributes that might be chosen to describe and recognize different orders. The Boolean properties are used to track the process status. In general, the properties of relationship types are unidirectional references to the participants. Here, the shopper, seller and delivery relationships are used to specify the customer, manufacturer and logistics of any business network instance determine the verifiable individuals or legal items with access to read a particular asset. Besides, the responsible property, the relationship to the partners, which is required to take action, is used to determine which one of the related parties can update each situation's asset.

We can define different collaborators of a collaborative business process as separate participants in the model file. It is better to define fewer participants in the code and specify their kinds using some enumerated types. Accordingly, we only require to define one participant in the order processing model file. This participant has a property with enumerated type, which can be an option of shopper, seller, or delivery. Furthermore, the participant definition includes some other properties to add the identification attributes

Table 3.1: Asset properties in the order processing model.

Property	Type
status	Enumerated
ID	String
name	String
description	ID
recieveOrder	Boolean
rejected	Boolean
accepted	Boolean
fillOrder	Boolean
sendInvioce	Boolean
makePayment	Boolean
acceptPayment	Boolean
shipOrder	Boolean
closeOrder	Boolean
shopper	Relationship
seller	Relationship
delivery	Relationship
responsible	Relationship

such as first name, last name, ID, company name, and Intra organization position. Similar to the asset resource, any participant resource requires to present an identifying field, and their unique assigned value distinguishes its instances for this field. Here, I selected the participant's ID property as the identifying field, but all the identification properties are arbitrary and may vary based on the content and obligations of the business processes.

In addition to the asset(s) and participant(s), we must declare the transactions in the model file. Generally, each transaction declaration involves its name, relationships to the assets and the participants who need to work with, and required properties holding new input values. Considering the collaborative business process's asset definition, I defined one transaction for each Boolean property to update its value. In other words, transactions are required for those elements of the BPMN process-flow that oblige parties' actions. Additionally, a unique transaction is also necessary to initiate new instances of the asset. Accordingly, I created nine transactions for their corresponding tasks and XOR gateway of the order processing model and one transaction to create a new asset instance.

I coded the definition of model file's declared transactions in transaction processor functions (smart

contracts). For a collaborative business process, I reflected the BPMN model objects' sequence in transaction processor functions such that each transaction can be performed if and only if all its required predecessors were already accomplished. For example, in the order processing model, the fillOrder task can be done if and only if the received order formerly was accepted. The Close Order task can also be executed only if the received order was rejected or both acceptPayment and shipOrder tasks were fulfilled successfully. Additionally, the smart contract does not allow you to execute a task's transaction more than once. Although the script file explains each transaction's details and their logical order, it does not answer this question: who can perform a specific transaction? I answer this question in the next section.

3.4.2 Business process access control

I defined access control rules for any collaborative business processes using the ACL inside the access control file. Regularly, these rules may contain CREATE, READ, UPDATE, and DELETE (CRUD) [108] actions overall resources specified in the model file for any participant or component of the business network or their instances. We can implement conditional access control rules, including Boolean JavaScript expressions, to accurately implement the business norms. Furthermore, according to the business model's content and requirements, different access control techniques such as RBAC and ABAC could be implemented. For the order processing business model (figure 3.1), I placed relevant access control rules to ensure the following principles:

- Only a customer can create a new instance of the order processing asset, which is a commodity request.
- Only the related shopper, seller, and delivery, who are separate parties of the collaborative business, can read the asset's created instance.
- The only one who can execute the corresponding transactions to the Receive Order, Accepted/Rejected, Fill Order, Send Invoice, Accept Payment, and Close Order objects are the asset instance's associated seller (or manufacturer).
- The related shopper of an asset instance is the only one who can execute the corresponding transaction to perform the Make Payment task.
- The related delivery can only execute the corresponding transaction to the asset instance's Ship Order task.

I assumed that the customer selects the intended delivery from all different options while creating a new asset. However, we can define it otherwise to cover other scenarios as well. For instance, the seller might be interested in working with some particular deliveries. Therefore, some implementation details still may vary from case to case.

3.4.3 Deployment and execution of the business network

I have examined a case study to determine the validity of the implemented approach and the accuracy of the deployed business network on blockchain.

This test was initiated with 28 participant instances, including twenty shoppers, five sellers, and three deliveries. I created 200 random asset instances (table 3.1), including shoppers, sellers, and deliveries. All valid and invalid transactions are defined for every step of the process flow, holding the asset instances as separate business processes. The invalid transactions include intra-process and interprocess transactions that do not follow the order processing model's rules. In opposition, the valid transactions conform to the designated BPMN model's logic from the start event to the end.

The test went through valid transactions for each asset to complete the business process and fulfill the closeOrder task. I designed the answer to the XOR gateway decision in the order processing's BPMN flow chart (figure 3.1) randomly so that each asset may include either true "rejected" or true "accepted" property (table 3.1) at the end. After each valid transaction, the test tried an invalid transaction. This invalid transaction could be a randomly intra-process or inter-processes transaction. Finally, the accuracy of the presented business network is ascertained in terms of the number of successful valid transactions and the number of failed invalid transactions. Table 3.2 outlines the statistics result of the test experiment.

Table 3.2: Business network test configuration and result.

Number of participant instances with shopper type	20
Number of participant instances with seller type	5
Number of participant instances with delivery type	3
Number of all asset instances	200
Number of asset instances with true "accepted" property	126
Number of asset instances with true "rejected" property	74
Number of all valid transactions	1430
Percentage of successful valid transaction	100%
Number of all intra-process invalid transactions	817
Number of all inter-processes invalid transactions	613
Percentage of failed invalid transactions	100%

3.5 Chapter summary

In this chapter, I discussed the implementation details of Order Processing in business process management in a trusted permissioned blockchain. The chapter contributes to the body of data trust as a case study

of establishing transparency and trust between multiple untrusted parties with the ability to track, monitor and verify every committed process. In summary, implementing the collaborative business processes on the permissioned blockchain extends the following benefits:

- In a permissioned blockchain, only authenticated users can join the network and read the state of the ledger
- Blockchain grants a distributed tamper-proof and shared ledger that makes the occurrence of any dishonesty almost impossible. Hence, different parties can trace the situation of the process and confirm its detail accurately.
- Presenting the consensus mechanism, the business parties do not need to trust any single entity or any third party anymore.
- Smart contracts can contribute to implementing the business process's logical sequences and equip business parties to do appropriate possible actions. Furthermore, smart contracts can be utilized to perform automatic transactions as well.
- According to the business model's specifications, any access control method such as discretionary or role-based access control can be implemented to define the access levels over parties' data, assets, transactions, and the process flow's data.

3.6 Contribution of the work presented in this chapter

This chapter studied data trust in an untrusted business process management, where multiple untrusted business parties can collaborate for a common purpose in a trusted environment. The applicability of the order processing execution as a real-word untrusted business process on the permissioned blockchain was investigated. A proof of concept based on Hyperledger Composer to confirm blockchain technology's effectiveness for the presented purpose was developed.

The contribution of this chapter is published in the following paper. I collaborated in system design, implementation and writing the paper.

Pourheidari, Vahid, **Sara Rouhani**, and Ralph Deters. "A case study of execution of untrusted business process on permissioned blockchain." In 2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), pp. 1588-1594. IEEE, 2018.

4 Sharing medical data using blockchain

Scattered medical data has been long-established in health care delivery systems. Medical data are usually recorded in private databases. As a result, patients' data are scattered across different healthcare institutions, even political jurisdictions (different provinces or states), since patients do not restrict themselves to a single doctor or clinic. This prevents collaborative healthcare treatment and access to patient data efficiently.

This issue originates from the initial wrong presumption that the electronic medical or health record (EHR and EMR) systems assume that patients receive medical services only from practitioners in one clinic or one political jurisdiction. Though, in practice, patients do not restrict themselves to a single doctor or hospital. The ability of these systems to interoperate and share medical data is deficient.

A survey in 2010 [1] has reported that the average U.S. patients have visited roughly 19 distinct medical institutes. This study did not reflect the broader health practitioner state, such as pharmacists, physiotherapists, optometrists. Patients also do not bind themselves to a single EHR zone, as they travel for leisure and work, and they relocate for long periods. Therefore, patients receive medical service from different institutes where that provider may not have access to the patient's medical records.

On the other hand, these systems are practitioner-centric. Patients lack the ability to control and transfer permission in managing their own data. Institute and government policies determine access to the patient data without involving the patients as real data owners. Therefore, even if the patients are willing to share their data, the procedure is prolonged.

Big Data in healthcare and medicine includes different types and large amounts of data generated from clinics or hospitals, such as electronic medical records, medical imaging and medical reports. It is often closely associated with doctors, patients and pharmacists and consequently researchers.

Medical data is the primary source of health and safety information for patients, making it more sensitive and private. Despite enormous investments by medical facilities and governments, medical services delivery and health management continue to be hampered by data inaccessibility, poor interoperability, inconsistent security of medical data assets, lack of privacy controls, and excluding patients from control access to their own data assets.

Studies [167] have shown that merely storing medical data to the cloud systems and managing them for sharing and analytic is not a secure option as cloud services regularly face internal and external security threats. Thus, outsourcing sensitive health data to the cloud platforms without additional security and privacy protection arrangement will add security risks.

In this chapter, I investigate blockchain potential as the infrastructure of a trusted system for sharing

Figure 4.1: MediChainTM components.

medical data directly through data owners, which are patients in this particular use case. By utilizing the decentralized nature of blockchain and automatic enforcement of smart contracts, we can develop a system that does not rely on third parties to share and manage healthcare data access permissions. We can design the system to define medical data as an asset belonging to the patient. They directly have the power to submit transactions and share their medical data with anyone who is a member of the system.

Although there are similar works in the scope of access control based on blockchain, MediChainTM is the first implementation of a functioning, real-world application based on Hyperledger Composer and Hyperledger Fabric to utilize a permissioned blockchain to address these problems in healthcare systems.

This chapter addresses these issues with a blockchain-based architecture that allows MediChainTM to be extensible and scalable. MedichainTM includes three main components represented in figure 4.1: a blockchain-based access control module, off-chain data storage, and a patient-centred mobile and web user interface. Medical data, such as X-ray images, are space demanding by nature. As blockchain ledger replicates on all peers, storing the original medical data into the ledger is not efficiently appropriate. In order to maintain system performance, all data assets such as diagnostic images, lab test results, prescriptions, treatment plans are encrypted and stored on a secure cloud-based repository. The hash of the asset's and URI (Uniform Resource Identifier) data assets are stored in MediChainTM permissioned blockchain. With the hash value, users can ensure that their data does not tamper. With URI, users can access data.

The system's patient-centric design aims to solve the trust issue for the patients to share their data. First, the patient will trust the system as long as they can control access to their data and transparently monitor who has access to it. Consequently, it is expected that patients are more inclined to share their data.

4.1 Early studies of using blockchain for medical data management

In early 2017, very few studies implemented a medical data sharing system based on blockchain, and my study is one of the earliest research on this topic. Furthermore, existing studies were limited to design and architecture solutions, and the implementation and system functionality was missing in those studies. However, it was essential to identify the characteristics, design patterns and architecture of existing studies.

Rosted et al. [139] proposes a patient-centric healthcare data sharing model using role-based and DAC

methods. In the PCHR model, patients have the ultimate authority in determining who can access their data.

Enigma [209] is a decentralized computation platform that considers privacy, computation, and data storage to achieve privacy and scalability. The platform splits data into unmeaning chunks, and each node has access to one chunk of data. Unlike blockchain, the data are not replicated and computed by every node. However, an external blockchain is employed to control the system, manage access control, and consider a tamper-proof database of events.

Zyskind et al. [210] employed blockchain to omit third-parties for personal data access management. As a result, users can control access to their data. They implemented the system based on the combination of off-chain storage and storing a pointer to encrypted data on the Bitcoin blockchain. The blockchain portion of the system handles sharing and querying the data.

Medrec [15] is another implementation of an access control system that has used blockchain technology. Medrec used Ethereum technology with some modifications in the mining process. The platform presents a reward-based mining method to motivate medical stakeholders to participate in the system and verify transactions as miners.

Xia et al.[184] introduced a blockchain framework for sharing electronic medical data stored in the cloud repositories. The proposed system is based on a permissioned blockchain; therefore, only authorized users would have access to the system by verifying their cryptographic keys. The performance testing based on comparing this blockchain and bitcoin blockchain confirms light and scalable design.

Cruz et al. [46] presented an implementation of role-based access control using the smart contracts and the challenge-response protocol based on the Ethereum blockchain. They designed a challenge-response protocol to authenticate the ownership of roles and verify users. RBAC-SC has focused on trans-organization access control; a user accesses one organization's service based on his or her role in another organization.

Guo et al. [69] employed blockchain for validating EHR records by using an attribute-based signature (ABS) method with various authorities. The evaluation result demonstrates that their system is robust against collusion attacks and preserves the privacy of users.

The presented system (MediChain) [142] is one of the earliest studies that implemented interactions between permissioned blockchain and off-chain cloud repository and indicated the proposed solution's feasibility.

4.2 Selecting the right blockchain platform

The few existing systems have used Ethereum blockchain to implement and analyze the proposed platform for data sharing using blockchain. In Ethereum, users must pay for every operation in the format of transactions. I selected the Hyperledger Fabric for developing the system. Figure 4.2 presents a representative Hyperledger Fabric-based application. Here are the main advantages of using Hyperledger Fabric:

1. Fabric is a mature blockchain platform, and users could have different access levels based on their

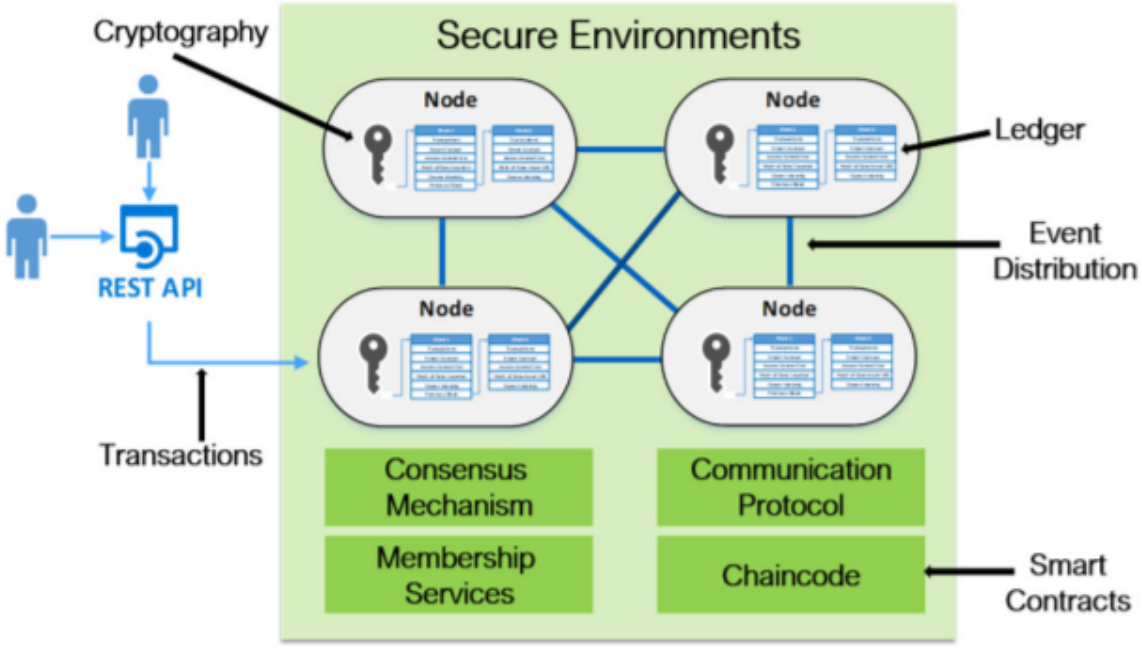

Figure 4.2: Application based on Hyperledger Fabric.

identity.

2. There is no cost of operations such as Gas in Ethereum blockchain, which reduces the overall system cost.

3. The capabilities to communicate through a different channel and its support for private data are desirable for the presented application if different organizations require private communication and smart contracts for exchanging medical data.

4.3 System implementation

The implemented system includes three main components: a permissioned blockchain, off-chain storage, and a patient-centric user experience accessed through a web app interface. Figure 4.3 provides an overview of the presented architecture as instantiated in the TrioovaTM [1] application. I describe each main component as follows.

I used Hyperledger Composer to create the Business Network Archive (BNA) that defines the characteristics and capabilities of MediChainTM. I deployed a runtime version of the BNA generated by Hyperledger Composer over the Hyperledger Fabric network. There are three primary files made with Hyperledger Composer: a model file, a script file, and the Access Control List (ACL) file.

[1] https://www.bioalberta.com/trioova

- Model File: In the model file, I defined the Participants, Assets, and Transactions supported by MediChainTM
- Participants: As a permissioned blockchain, all users are authenticated by a third party before giving access to the blockchain components. When a user is authenticated, the Fabric membership service assigns a blockchain ID to the Fabric wallet. There are three types of participants in the system: patient, caregiver, and health practitioner. An individual can register as one, two, or all three and unique IDs are assigned to each registration. Each participant type can be a data owner and a data requester. Data requesters are authenticated users who can submit transactions that request access to particular data owners' data assets.
- Assets: In the TrioovaTM application, assets are patients' medical data. Initially, each asset is assumed to be a private asset. The owner is the only participant who has access to the asset. The owner of the data has the right to share a data asset with others via transactions.
- Transactions: Any request for a change to asset access starts with a transaction initiated by a data requester. The requests are managed in a publish-subscribe fashion.
- I implemented smart contracts to establish the patient and caregiver to share and access healthcare data using blockchain without third-parties interventions.
- Script File (Smart contracts): The smart contract provides transaction details functionality. I implemented the conditions and requirements of each transaction in the script file. The web interface enables the user to specify input parameters of transactions and trigger them. For example, we can design a transaction in the smart contract to provide time-based access to a data asset. The data requester provides the asset's identity and the proposed length of time (or date, etc.) for accessing the data asset. The data owner can agree to these terms or suggest modifications. When the conditions are met, the validator peers confirm the transaction.
- Access Control List: Hyperledger Composer provides complex condition language that can be used to specify default policies and actions for the runtime BNA. In this file, I implemented the users who have default access permissions to the patient data, such as the family doctor or a close relative.
- Events and queries: I used event and query components in Hyperledger Composer to implement the proper events and queries. After all of the initial entities and transactions have been set up, the logging and querying methods will be used to track all submitted transactions. Any data owner can track and monitor any request. This provides the auditing capability.
- Off-chain storage: Recording the medical data directly on the blockchain is not an appropriate solution for two main reasons. Medical data, such as medical images (X-rays and others), are large by nature. Storing such large data increases the chain's size significantly. Besides the performance issue, storage

would be another concern because, in blockchain, every peer maintains a copy of the ledger. Therefore, I kept the data in a secure decentralized off-chain system. In proof of concept implementation, I used Google Drive to store the medical data. (However, we can integrate any other cloud storage solution with the application.) I stored the data location references as a part of the medical data asset on the ledger.

- Web interface: TrioovaTM is the web interface that I implemented using AngularJS to access the MediChainTM capabilities. The core principle is that the patient is always at the center of a circle of care. Patients determine who to include in their care circle and, at a granular level, what data to be shared with each. For patients who need or want a trusted party's assistance, the circle expands to include a caregiver that can act on the patient's behalf.

The proposed method securely links on-chain and off-chain data without incurring the restrictive computational and storage loads inherent in most blockchain architectures. By utilizing smart contracts, the patients, as data owners or their designated person (e.g. a caregiver), can easily manage access permissions to each of their data assets. I developed a web application to interact with the application easily accessible to patients, caregivers, and health practitioners. The proposed platform can also be utilized to find relevant health practitioners, request appointments, and manage care plans across multiple health practitioners.

I implemented access control transactions using smart contracts, in which patients, as the owners of healthcare data, can share their data as a part of an asset belonging to them directly. The patient is the only one that can submit an access grant transaction unless the patient adds another user to delegate the permission to do so. I also implemented a role-based access control model for providing default access permissions to family doctors and maybe close relatives of the patients. I used Hyperledger Composer for implementing the application on top of Hyperledger Fabric.

4.4 System workflow

Figure 4.4 illustrates the workflow of the system. The process is initiated by a patient who receives a medical service in a medical clinic. The patient could request to add the results, including medical images, practitioner notes, and prescriptions, to Medichain. The patient's original data is stored securely in Medichain cloud-based storage (off-chain storage). A transaction is executed, and the reference to the data location plus the hash of the original data will be stored as a part of the patient's asset on the blockchain. If the patient is already assigned a practitioner, the practitioner gets an automatic notification that there is data that just added to your patient asset. Then, the practitioner must send an access request to the patient. The patient receives the request and can define the access terms and submit the access request's result. The result of the access request will store on the Medichain blockchain. If the patient approves the access request, the caregiver gets a notification and can access the patient medical data.

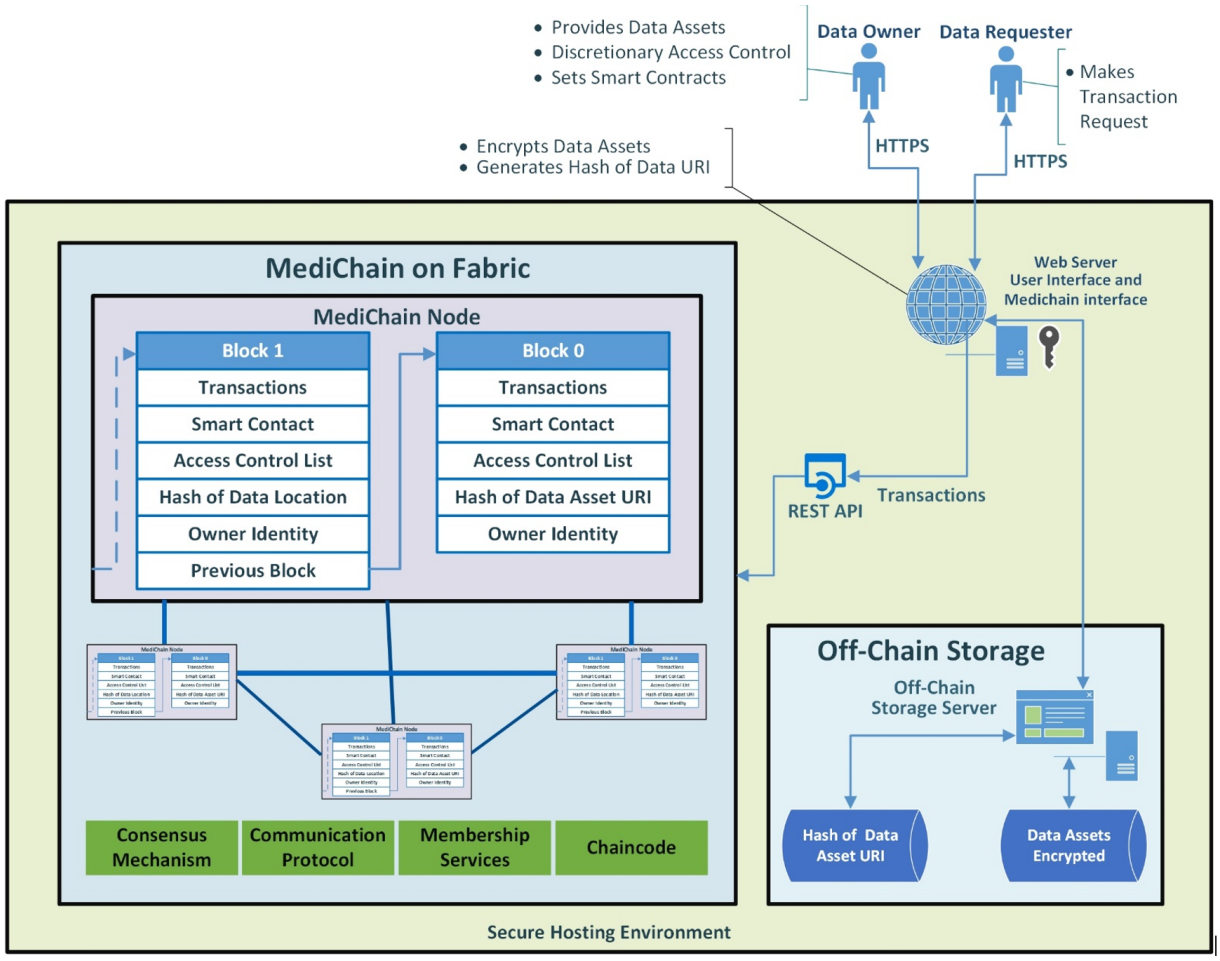

Figure 4.3: MediChainTM architecture.

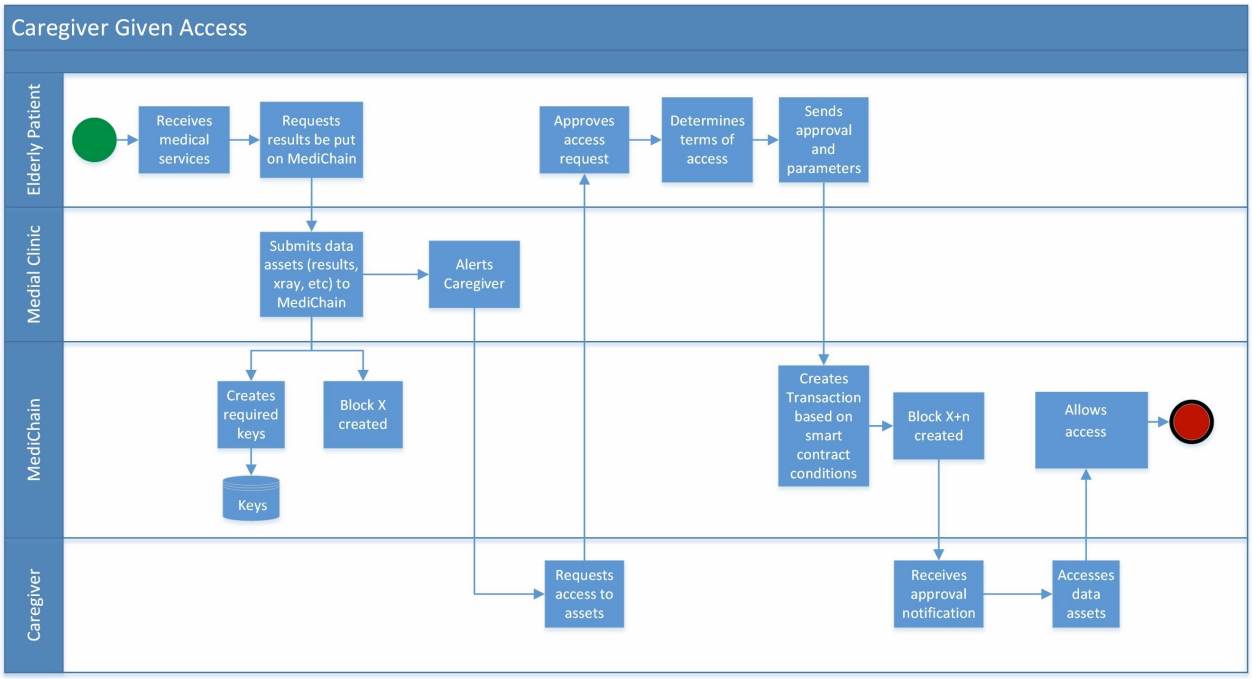

Figure 4.4: MediChainTM workflow.

4.5 Chapter summary

In this chapter, I presented a framework for sharing medical data with patient-centric design. The patient, as the owner of medical data, has full control over access to their data. They decide on access requests, execute the access decision transactions and audit the users who access their data.

The system facilitates the efficient exchange of health data between patients, caregivers, and even people with research purposes while simultaneously enabling a secure protocol for health data storage. This system provides the preferred ability to manage access to medical data, enhanced security of those data, and consequently engage patients to share their data. The current architecture can be extended to be generalized to any use case requiring the management of large digital assets such as music, visual arts, and documents.

The chapter contributes to the body of data trust as a case study of trusted medical data stewardship.

4.6 Contribution of the work presented in this chapter

This chapter contributed to the body of knowledge by presenting a patient-centric system for access management and medical data sharing. The system utilized blockchain's smart contract to manage access permissions under the control of patients. A proof of concept to evaluate the proposed solution was implemented.

This study is the result of my Mitacs internship with Trioova company. The contribution of this study has been published in the following IEEE conference.

Rouhani, Sara, Luke Butterworth, Adam D. Simmons, Darryl G. Humphery, and Ralph Deters.

"MediChain TM: a secure decentralized medical data asset management system." In 2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), pp. 1533-1538. IEEE, 2018, doi: 10.1109/Cybermatics_2018.2018.00258.

5 Physical access control system

In the previous chapter, I presented a decentralized access control and data sharing for medical data. This chapter introduces a decentralized access control mechanism for another application domain. This chapter presents my study focusing on access control for the software layer of the physical access control system.

There are two types of access control systems, logical access control and physical access control. The logical access control system manages access permissions to the logical resources such as data. Physical access control limits access to physical places such as buildings, rooms, and network equipment. Most of the studies on access control systems have focused on logical access control. While physical access control systems are also very important, and there is potential to enhance their security and reliability.

Physical access control systems use keys, badges, and fingerprints to authenticate users and grant access. Behind these hardware infrastructures, there is a software layer that manages access permissions. The software layer handles users' access control and keeps track of who is coming and going in restricted areas. Typically the software system in physical access control systems is implemented centralized, and organizations must rely on third parties for implementing and maintaining their physical access control system.

As I discussed earlier, there are multiple security and reliability concerns with centralized solutions, such as single point of failure, availability, data tampering, the presence of third parties and hacking attacks. Besides the security vulnerabilities that arise from involving a third party, the logging approach in centralized implementation is not entirely trustable. There is always the chance of malicious behaviour and erasing all available evidence.

Blockchain provides a reliable history of all transactions, which are metadata related to access requests and responses in the current case study. Moreover, with smart contract automation, the system can operate directly by parties without requiring third parties.

In this chapter, I discuss the blockchain's potential as an infrastructure for the software layer in the physical access control systems. I utilize blockchain's immutability and tamper-proof features for access control systems by accessing an auditable transaction trail.

There are two main advantages of using blockchain for physical access control systems. First, the history of all recorded transactions stored on a trusted tamper-proof database significantly reduces the chance of inner organization malicious behaviour. This trust comes from the secure and solid logic behind the cryptographic nature of the blockchain. Second, using smart contracts for access permissions management eliminates the need for third parties for system operation and maintenance. Authorized users can grant or revoke access permissions directly.

In this chapter, I outline the access control application [145] implemented using Hyperledger Composer [1] and Hyperledger Fabric [2]. Figure 5.1 shows how the software layer of a physical access control system interacts with the blockchain to request data from the blockchain and read access permissions.

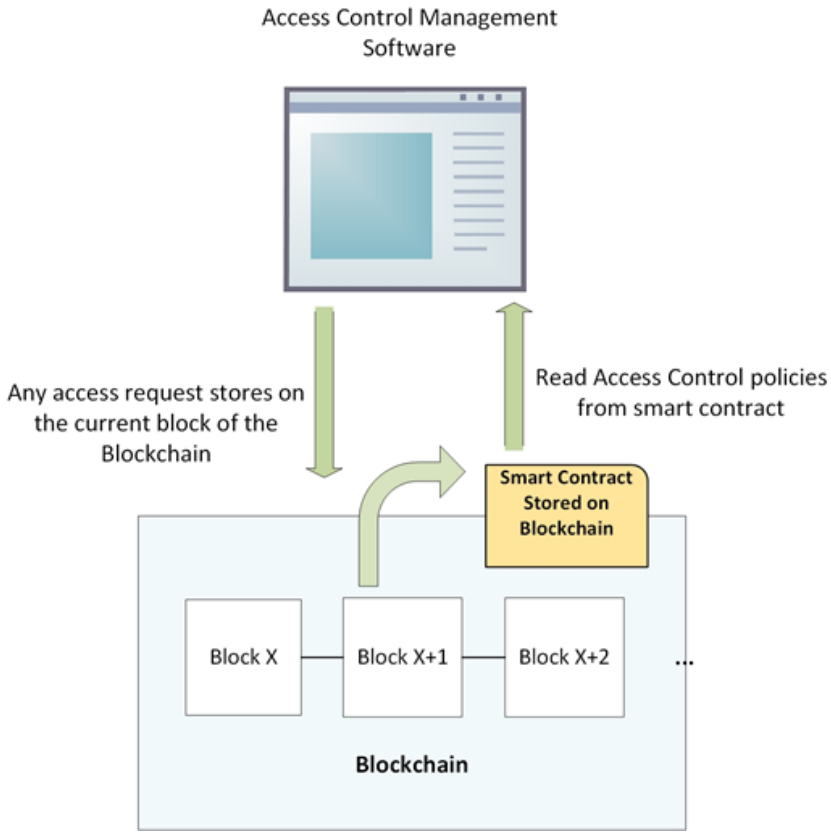

Figure 5.1: The interaction of access management software with blockchain.

5.1 Access control method

Most of the physical access control systems are designed based on the role-based access control model. Role-based access control is easy to implement and operate. After reviewing various access control models and considering the blockchain capabilities, I developed the combination of two access control methods, role-based access control method and Rule-based access control method.

The Role-based access control method is applied to regulate users' default access permissions based on their roles in the organization. I also implemented rules using smart contracts, and they determine who has the privilege to change other users' access permissions. The authorized users can instantly modify users' access permissions by submitting the related transaction implemented in smart contracts.

[1] https://hyperledger.github.io/composer/latest/
[2] https://www.hyperledger.org/projects/fabric

I used the ACL module of Hyperledger composer to implement the role-based access control method to set participants' default access permissions. Each participant denotes a different role in the system. For dynamic or rule-based access control, an authorized user can change other users' access permissions by submitting the relevant transactions: grantAccess and revokeAccess. The transaction is only accessible to the predefined users. It invokes the related smart contract, and based on predefined rules, the default access permission of the intended user alters. All these transactions' results and logs are recorded permanently in the blockchain, and no one can remove or change them.

5.2 System implementation

I employed blockchain as a decentralized database for storing access requests and access permissions. Users send access requests through application API. There are two parameters involved in access decisions. First, the user role in the system determines the default access permissions. Second, despite the users' roles, their default access permissions can change directly by high privileged users (such as directors) and the rules and conditions defined in smart contracts.

Hyperledger composer is a framework to implement blockchain applications and deploy them on Hyperledger Fabric. All components in composer modular are packed as one archive file and deployed on the Hyperledger Fabric blockchain. The composer module includes a model file, ACL file, Query file, and a set of Scripts. Scripts in Javascript language are smart contracts.

In the model module, I outline Participants, Assets, Transactions, and Events. All participants in the system, such as systems administrators who define the access control policies and other authenticated users who need access to physical places, have been designated as participants in the model file. Defining participants include the unique participant key (such as user ID) and other attributes. I defined the physical places as data assets in the system. I implemented access to physical places through transaction processing functions or smart contracts and access rules defined in ACL file. For example, in the application, I considered a unique string that refers to the input door of physical places secured by a physical access control device. It could incorporate any other related attribute or map to any specific participant who can manage access policies over it.

Authenticated users submit transactions to change the default access permission dynamically. I defined the transactions' attributes in the model module and their processing functions (smart contracts) in the JavaScript file.

There are three transactions implemented for manage access permissions. One transaction for granting access permissions, one transaction for revoking a user default access permission. Additionally, there is a transaction to delegate authority to other users. In this way, they can submit the grant access or revoke access transactions to physical places for the other users. I wrote the default access control policies in the Access Control Language (ACL) module. These policies include the following items:

Table 5.1: ACL notations.

Notation	Meaning
P	Set of Participants
R	Set of Resources
C	Set of Conditions
A	Set of Actions (Create, Read, Update and Delete)
O	Set of Operations (Allow and Deny)

- Participants with specific roles have access to which resources by default
- Participants with specific roles can send transactions
- System access permissions
- System administrator role access permissions
- Participants with specific roles access permissions to read historian records

I defined five sets of Participants, Resources, Conditions, Actions, and Operations in the composer ACL file. Table 5.1 explains the meaning of each notation. In the basic model, users can access the resources based on their predefined roles in the system.

In the model file, I defined different types of participants to reflect different roles in the system. These participants have access to resources by default based on their roles and conditions.

$Ur \subseteq U \times r$ is the set of user-role assignments and $rR \subseteq r \times R$ is the set of role-resource assignments. A user u have access to the resource r if and only if there is a participant p defined in the model file and $(u, p) \in Ur$ and $(p, r) \in rR$ and condition c is met. A privileged user submits a transaction to change another user's access in the dynamic access modification model. This is separate from static ACL module definitions.

Events can be implemented as part of the model file and embedded in transactions. Then, we can subscribe events to an external application. For example, in this system, I assume several continuous efforts to access a physical place. As a result, the system triggers an event to an external application to apply necessary safety considerations such as triggering intrusion alarms to deter unauthorized access.

I implemented queries in the queries file to query transactions' logs. Log entries demonstrate the results of the transactions that have been fired from transactional processing functions. Composer Transaction Processor functions are part of JavaScript files. These JavaScript functions translate to fabric chainCode (equivalent to smart contracts). They define the logic of every transaction and the conditions that need to meet for executing this logic. The Transaction Processor Functions are automatically called through smart contracts invocation when an authorized user submits the relevant transaction. Figure 5.2 explains the access permission process based on the GrantAccess transaction.

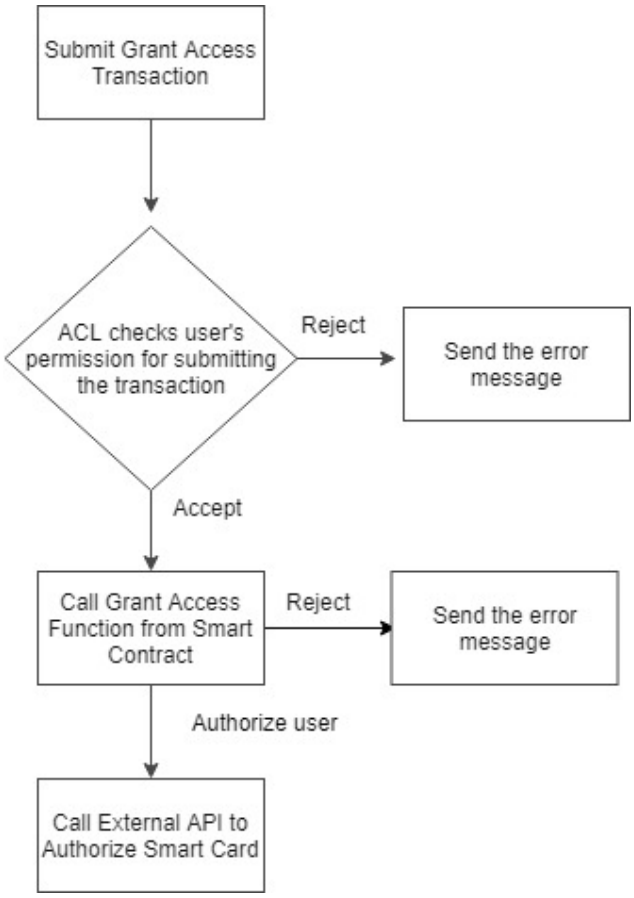

Figure 5.2: Access permission procedure.

Hyperledger Composer's Identity Management is responsible for creating a participant and issuing an identity to that participant. Composer employs the concept of cards inspired by real-world ID cards. An ID card is an access card to the blockchain network, and it includes three parts, 1) the data related to the identity, 2) a connection profile, and 3) a certificate for chain access. The connection profile is used to connect the Composer's network to the runtime Fabric. These ID cards can map to real-world physical access control smart cards to access physical places.

The query file incorporates the declaration of queries. Queries are based on tamper-proof data. We can be sure that no one has altered or deleted the data. Hyperledger Composer presents a Historian record feature. It records successful transactions with details, including transactions' information and participants who submitted the transactions. Figure 5.3 shows the historian transaction details.

Figure 5.4 shows the presented application's architecture model based on Hyperledger Composer and Hyperledger Fabric. The system comprises a model, smart contract (JavaScript code), access control, and query components. All these modules are packed into a business network archive (bna file). This archive file (.bna) later deploys on Hyperledger Fabric runtime Blockchain, and afterward, users can interact with the blockchain through JavaScript API and the application's user interface.

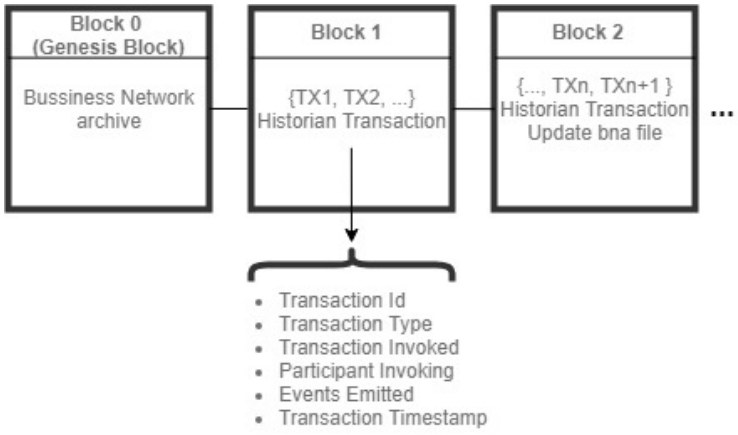

Figure 5.3: Historian transaction.

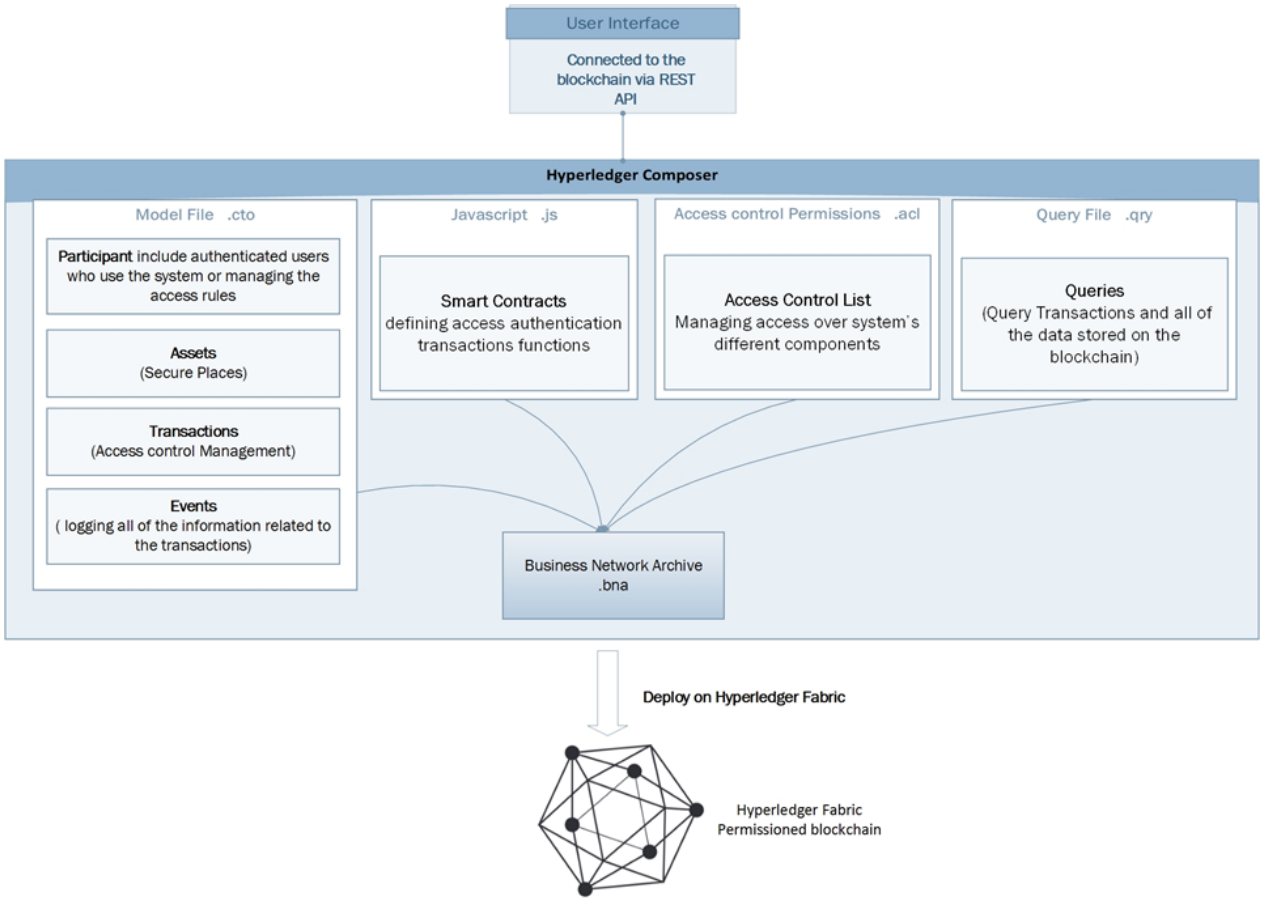

Figure 5.4: System architecture.

Name	Succ	Fail	Send Rate (TPS)	Max Latency (s)	Min Latency (s)	Avg Latency (s)	Throughput (TPS)
createAccessRule	500	0	10.0	1.75	0.54	1.08	9.9
queryAccess	500	0	10.0	0.04	0.01	0.02	10.0
queryACL	500	0	10.0	0.24	0.09	0.14	10.0

Figure 5.5: Performance result.

5.3 System performance analysis

I used Hyperledger Caliper [3] to generate the workload and analyze the performance of transactions. Hyperledger Caliper is a blockchain benchmark tool that allows users to measure the performance of their blockchain implementations. The system configuration used for testing the system includes 32GB memory (RAM) and Core(TM) i7-6700 CPU.

I implemented a test module to generate workloads based on implemented transactions. Three main functions of the test module include init(), run(), and end(). The init() function handles the initialization phase of the test. The run() function generates workloads and submits multiple transactions repeatedly in an asynchronous way. The end() function finalizes the end of the test.

As I explained in the previous chapter, in the model file, I mapped different participants to different roles in the system. Each role has a different level of access, such as managers and employees. I defined the physical places as assets. I categorized data assets into various departments, so access to each department is controlled by a different user (for example, the department's director). I defined user (participant) instances and asset instances in the init() function. Afterward, test Transactions were fired through the run() function.

Figure 5.5 shows the result of 500 transactions based on three different transactions, createAccessRule, queryAccess, and queryACL. The createAccessRule is a transaction that adds a new access rule to the ledger. It performs the write operation on the ledger, and its average latency is 1.08 seconds. The queryAccess and query ACL perform read operation from the ledger. The queryAccess transaction queries the access permissions for a single user, so it is a relatively faster transaction with an average latency of 0.02 seconds. The queryAcl transaction queries all access control lists for all users in the system. Compared to the queryAccess transaction, it is slower with an average latency of 0.14 second, but still faster than the createAccessRule transaction because it queries the ledger because it does not update the state of the ledger.

The test result indicates that the presented application has worked a hundred percent correctly based on a predefined use case. I tested the accuracy of the application in two phases. First, I checked users' access permission based on their roles to examine the accuracy of the ACL module. Second, I tested the smart contracts functionality by simulating a test in which authorized users (such as managers) change other users' access levels directly through invoking the related transactions implemented in the smart contract.

[3] https://www.hyperledger.org/projects/caliper

TYPE	NAME	Memory(max)	Memory(avg)	CPU(max)	CPU(avg)	Traffic In	Traffic Out
Process	node bench-client.js(avg)	-	-	NaN%	NaN%	-	-
Docker	dev-peer0.org1.example.co...0.1.0	125.5MB	121.7MB	101.01%	20.61%	4.5MB	3.7MB
Docker	dev-peer0.org2.example.co...0.1.0	121.4MB	115.1MB	102.06%	21.12%	4.5MB	3.6MB
Docker	peer0.org1.example.com	368.4MB	343.5MB	17.73%	11.75%	16.6MB	32.5MB
Docker	peer0.org2.example.com	359.6MB	333.3MB	17.87%	11.66%	16.5MB	32.7MB
Docker	couchdb.org1.example.com	114.5MB	109.2MB	54.88%	32.39%	4.5MB	8.1MB
Docker	couchdb.org2.example.com	117.2MB	110.9MB	53.61%	32.86%	4.5MB	8.1MB
Docker	orderer.example.com	17.7MB	14.7MB	3.64%	1.85%	3.9MB	7.8MB
Docker	ca.org1.example.com	5.3MB	5.3MB	4.37%	0.20%	4.6KB	3.6KB
Docker	ca.org2.example.com	7.3MB	7.3MB	0.00%	0.00%	1.9KB	0B

Figure 5.6: Resource consumption.

Figure 5.6 shows the result of the resource consumption of two organizations that each one runs one peer (peer0.org1.example.com and peer0.org2.example.com) and one orderer (orderer.example.com) in the Hyperledger Fabric network.

5.4 Chapter summary

In centralized systems, access to resources is controlled by trusted parties such as system administrators who have full control of the system's data and actions. As a result, centralized solutions are always in danger of security attacks and trust issues. This chapter presented a novel application to implement an access control system using permissioned blockchain for physical access control systems. By utilizing Hyperledger Fabric and Hyperledger Composer potential, I have implemented a dynamic access control application for managing access permissions on physical places. The system is trustable since it is protected from unwanted tampering.

The system provides a reliable transactions log to query, and it could be accessible by authenticated and authorized users. The transaction logs of access requests and access permissions can later be utilized in terms of non-repudiation. The analysis reports extracted by Hyperledger Caliper indicates system stability and accuracy, including a hundred percent successful transaction rate.

5.5 Contribution of the work presented in this chapter

This chapter proposed a novel application for employing blockchain in physical access control systems. It improves the security and reliability of software layers in physical access control systems by managing access permissions through smart contracts and audit access requests from the tamper-proof ledger. The reliable and secure infrastructure reduces the chance of inner organization malicious behaviour. The system was

implemented using Hyperledger Composer and evaluated by Hyperledger Caliper.

The contribution of this chapter has been published in the following paper:

Rouhani, Sara , Vahid Pourheidari, and Ralph Deters. "Physical access control management system based on permissioned blockchain." In 2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), pp. 1078-1083, IEEE, 2018, doi: 10.1109/Cybermatics_2018.2018.00198.

6 Decentralized Attribute based access control system

Auditing is one of the essential controls in systems security. Auditing is the action of tracking all access attempts, including both legitimate and illegal access attempts. Keeping track of legitimate access attempts helps with non-repudiation, and keeping track of illegal access attempts helps identify potential threats.

Auditability is also one of the key characteristics of blockchain. Blockchain provides a trustable history of all transactions. Blockchain can employ smart contracts to store access control policies, make access decisions and store the result of access attempts. Then, at any point in the future, all the access attempts toward a particular resource can be queried from blockchain, and it can be used as an authentic proof for non-repudiation, or it can be studied for further analysis to identify possible threats.

Besides, blockchain presents other beneficial features that are desirable for access control systems, such as immutability and transparency. For example, suppose a malicious system administrator changes a policy to grant or deny someone access. In that case, it will be recorded on the blockchain, and it is not possible to delete the trace of updates on policies from the blockchain. Each policy has a history of changes applied in the policy that can be queried by permissioned users in permissioned blockchain. However, we can avoid such a problem by configuring smart contracts so that authenticated parties must approve any change in access control policies before execution.

Many studies have investigated blockchain as a back-end infrastructure for the distributed access control system. However, most of the prior works in this area are domain-specific. It means their designed access control solutions for a particular domain, such as healthcare data or IoT data. Besides, most of these studies lack the details of implementation and performance analysis. As a result, it remains unclear if a blockchain can be the basis for access management at a large scale.

In this chapter, I propose a complete solution for implementing an ABAC system focusing on policy-based architecture based on Hyperledger Fabric permissioned blockchain. The research contributions are summarized as follows:

- All access control components including AM, PDP, Policy Administration Point (PAP) are implemented as smart contracts (*or Chaincode*) and they are operated by Hyperledger Fabric.
- I utilize Hyperledger Fabric as Policy Information Point (PIP) to provide accountability through the immutable, traceable transaction history offered by Fabric ledger.
- I present a specific use case of digital libraries to represent the system operation modelling.
- I carry out multiple experiments, I present the performance analysis of the presented access control

application using Hyperledger Caliper [1] based on various configurations, including different databases and consensus methods.

- I also present a performance analysis results conducting from a comprehensive comparison between existence configurations based on Hyperledger Fabric.

6.1 Attribute based access control

ABAC [80] is logical access control that comprises access control lists, role-based access control, and its own method for providing access based on the evaluation of attributes [79, 192]. ABAC controls access to the system resources by evaluating policies (system rules) against entities' attributes, including subject, object, and environmental attributes. Attributes are characteristics of the subjects (users) and protected objects (resources). The environment conditions as the environment's attributes can also be taken into account for ABAC decision making.

ABAC is a flexible and fine-grained mechanism that is also capable of enforcing the other three methods. Distributed systems also adopted ABAC as they require federation and autonomy control among coordinated systems, and ABAC enables granular and meta attribute capabilities that support privilege delegation in a distributed application [78].

ABAC has some advantages over other access control models as, (a) It can provide fine-grained and flexible access control because it allows an arbitrary number of attributes in access control decisions; (b) The implementation of complex policies is simple and applicable; and (c) It can provide dynamic and effective access control decisions by involving environmental attributes to decision making.

Hyperledger Fabric has integrated the ABAC mechanism, so it is possible to build permission groups for access control by checking members' attributes. However, access control parameters and permission groups have to be predefined. It is not suitable for applications that require dynamic and flexible access control.

6.1.1 Policy based architecture

XACML [12] introduces a policy-based architecture for the specification and enforcement of access control policies. The architecture comprises the following components.

- *Client*: the device that requests access to a resource, possibly on behalf of a user.
- *Policy Enforcement Point (PEP)*: the network device on which access decisions are carried out. PEP serves as the gatekeeper to the intended resource.
- *Policy Information Point (PIP)*: the repository that holds information (attributes) about the client and provides this information to the PDP.

[1] https://www.hyperledger.org/projects/caliper

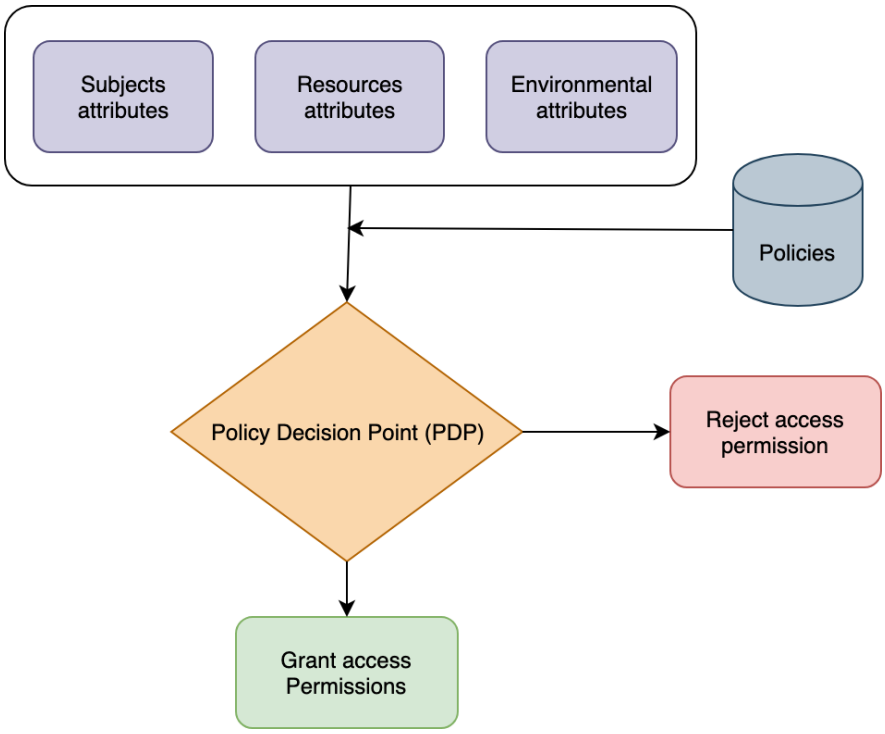

Figure 6.1: Attribute based access control.

- *Policy Decision Point (PDP)*: the program that decides to allow or deny the client access to the resource.
- *Policy Administration Point (PAP)*: the component that is responsible for managing access control policies.
- *Accounting or Auditing*: The component that is responsible for tracking access attempts.

Figure 6.2 illustrates how these components interact with the client and each other. A client sends an access request, and Policy Enforcement Point (PEP) forwards the request to Policy Decision Point (PDP). Policy Decision Point (PDP) queries the related policy and attributes from Policy Administration Point (PAP) and Policy Information Point (PIP). After receiving the required information, Policy Decision Point (PDP) assesses the access decision and sends the result to Policy Enforcement Point (PEP) for enforcing access decisions.

6.2 System model

Centralized access control systems suffer from various problems such as: (a) the risk of privacy leakage, and (b) the risk of a single point of failure; (c) interoperability issues; (d) unreliability of the access control system, and (e) the presence of third parties.

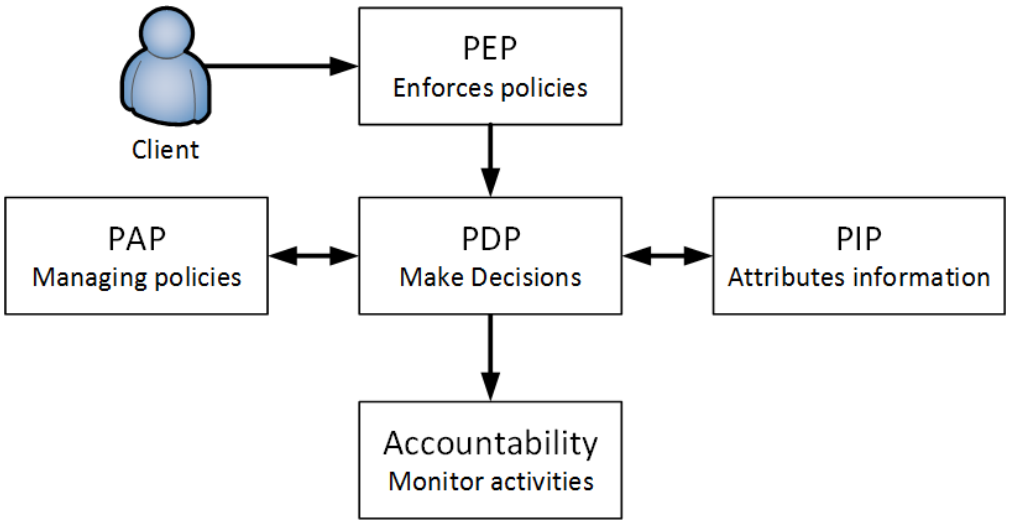

Figure 6.2: ABAC and policy based architecture [12].

An access control system can utilize blockchain technology to address these problems. The decentralized nature of the blockchain resolves the problem of a single point of failure. Cryptographic methods ensure the reliability of the ledger. Consensus mechanisms ensure that the state of the ledger is valid and it is the same for every participant. Smart contracts allow monitoring and enforcement of sophisticated access control decisions. Also, with automatic enforcement, they can address privacy issues.

This solution empowers both resource owners and subjects (typically access requesters). The details of each granted or revoked access permission can be queried from the ledger, including the policies that have been applied by the smart contract, the attribute values and the time of access request.

In practice, under no circumstances, resource owners do not deny access to a resource by a rightful requester. On the other hand, the resource owners leverage provided audit trails while ensuring that no user has subverted the system.

To provide a solution to these problems, I have employed Hyperledger Fabric as the blockchain infrastructure. In the blockchain, there have to be at least two endorsing nodes belonging to different organizations. These nodes are responsible for executing smart policies. Clients would be the systems that use this system, as depicted in figure 6.3.

The workflow of the users remains unaltered. The only difference is that the authorization requests are now mediated through one or more nodes representing the given system, as the evaluation of access control policies could be not trusted for the subject of the request, who instead requests invalid access to resources.

In order to protect users' privacy, Hyperledger Fabric provides private data features to protect sensitive users' data. I have utilized this feature to represent the attributes required for access permissions based on the organizations' defined policies. The private data is hashed, and then it will be endorsed and ordered like other data. Finally, the chaincode writes hashed data on the ledgers of every peer. However, only organizations that require these private attributes to give access permission have access to them. Using Zero-knowledge

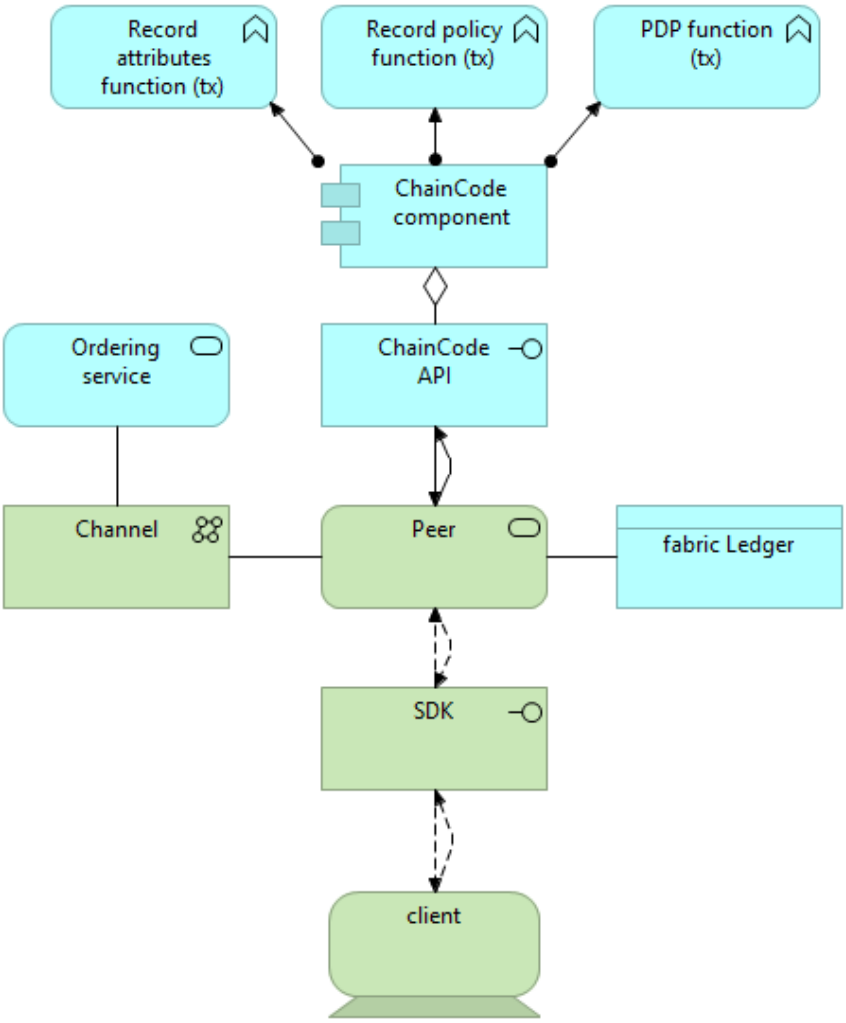

Figure 6.3: High level system architecture using Archimate.

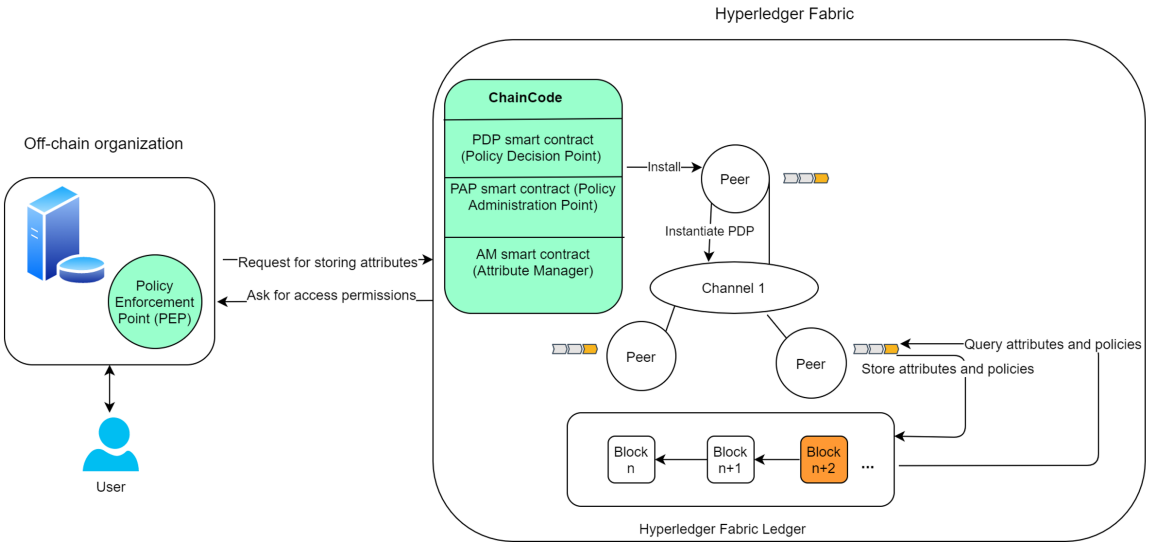

Figure 6.4: Blockchain-based access control system architecture [141].

proofs Zero Knowledge Proofs (ZKP) is another alternative that can be applied for highly private-preserving case studies that require the protection of all users' attributes from all access providers and data owners. However, ZKP requires additional time and computational resources compared to the presented solution based on Hyperledger Fabric private data feature.

6.3 System architecture

In the presented solution, as depicted in Figure 6.4, the blockchain acts as a mediator between the entity that requests access to a specific resource and the entity that manages that resource. The system includes two main components. The first component is an off-chain system that relies on a permissioned blockchain to store its access control attributes and query access permissions. The second component is a permissioned blockchain that manages different access control components through smart contracts and stores the data on a tamper-proof ledger.

The three main smart contracts are PIP contract, PAP smart contract and the PDP contract. Subject (users) and resource (objects) attributes are stored in a JavaScript Object Notation (JSON) data format through the PIP contract. The PIP is also responsible for checking write conflicts and updating attributes. Policies are also recorded in the system as JSON data format, and PAP contract is responsible for managing policies and updating policies. We can implement the system to work with multiple PAP run by different organizations. Even if there is only one PAP, the transparency offered by this solution distributes the trust and the responsibility of these access policies. PDP contract evaluates policies to make an access decision. Figure 6.5 shows the details of the implemented smart contracts.

The pseudocodes of transactions implemented in smart contracts are presented in algorithms 1-4. Algorithm 1 shows how we have stored the list of resources and subjects attributes on the ledger. Algorithm 2 presents how we have recorded policies on the ledger. Algorithm 3 explains how we query policies and the history of changes to the policies from the ledger. Algorithm 4 shows the detailed implementation of the PDP transaction, which includes querying the relevant attributes and policies from the ledger and making a decision based on their values.

Algorithm 1: Record attributes transaction

Input: attributeKey, attributes (subject or resource attributes)

Result: add attributes on the ledger

attributeKey (subject or resource key) ← getState(attributeKey)

if *subjectKey* **then**
| throw error 'attributekey' is already exist you can update the attributes using Update transaction

end

putState(attributeKey, attributes)

Algorithm 2: Record polices transaction

Input: policyKey, policy

Result: add policies on the ledger

policy ← getState(policyKey)

if *policyKey* **then**
| throw error 'policyKey' is already exist you can update the policy using UpdatePolicy transaction

end

putState(policyKey, policy)

Algorithm 3: Query and trace policies

Input: policyKey

Result: query policies info

policyBytes ← getState(policyKey)

if *!policyBytes* **then**
| throw error 'policyKey' does not exist

end

policy ← JSON(policyBytes)

resultsIterator ← getHistoryForKey(policyKey)

policyHistory ← allResults(resultsIterator)

After smart contracts evaluate SPs against their respective attributes, the PDP returns its decision to the PEP. This process allows a decoupling between users and the blockchain administration (as users do not

Algorithm 4: Policy Decision Point (PDP) transaction

Input: subjectKey, resourceKey, policyKey, rule

Result: permit

subjectBytes ← getState(subjectKey)

resourceBytes ← getState(subjectKey)

policyBytes ← getState(subjectKey)

subjectAttributes ← JSON(subjectBytes)

if *!subjectBytes OR subjectBytes.length == 0* **then**
| throw error subject's attributes does not exist

end

resourceAttributes ← JSON(resourceBytes)

if *!resourceBytes OR resourceBytes.length == 0* **then**
| throw error resource's attributes does not exist

end

policy ← JSON(policyBytes)

if *!policyBytes OR policyBytes.length == 0* **then**
| throw error policy does not exist

end

ABAC = Abac(policy);

permit = ABAC.enforce(rule, subjectAttributes, resourceAttributes)

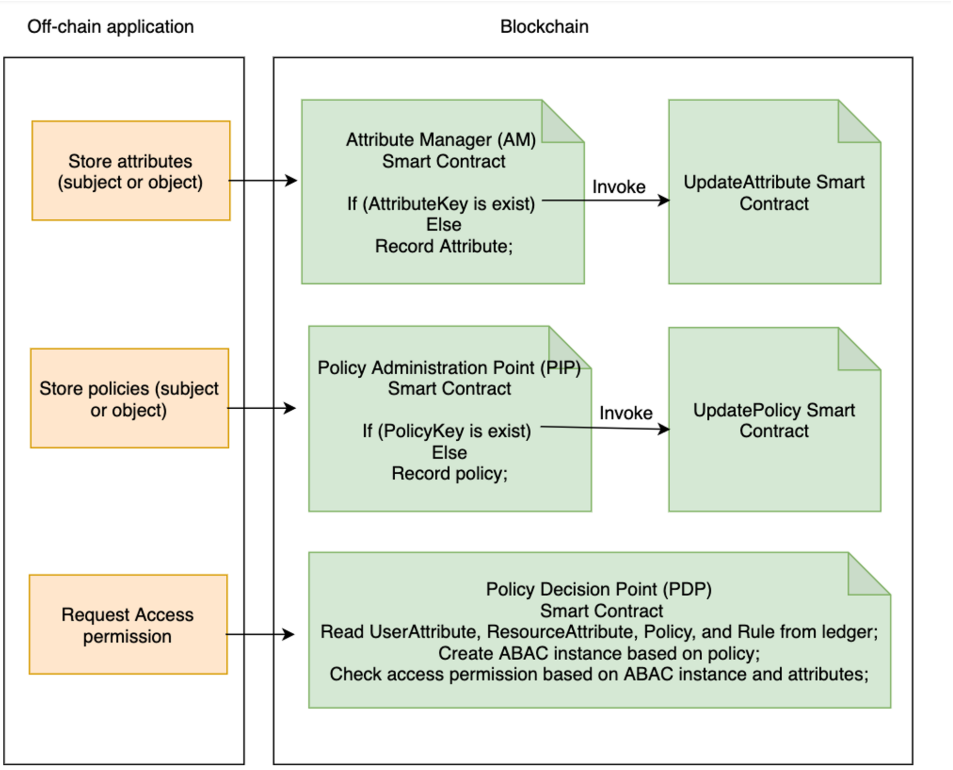

Figure 6.5: Smart contracts architecture.

need to have a node on the blockchain, which is desirable).

The presented solution logs all access requests in an immutable ledger and provides a framework to control all access controls concerning the network participants. Nodes from the private network can access the blockchain, check transactions' history, and audit the history of access requests and results. Automatic auditing techniques can be developed by analyzing the history of access request transactions. I provide a fine-grained access control solution that enforces access validation through blockchain-based service providers.

6.4 Case study

A digital library is a collection of documents in an organized electronic form that allows users to access them online. A highly dynamic user population and the numerous collection of digital libraries' resources require a fine-grained, dynamic access control method such as ABAC [4]. It requires that access policies specified based on users' attributes and characteristics rather than users' roles in the system.

In this section, a case study is selected for access control in digital libraries to illustrate and explain the application of the presented ABAC system.

Every subject is stored with a subject ID (SID), and the set of its attributes and values, and the same for Objects attributes. Attribute ID is a required field to store attributes in the Hyperledger Fabric database and later retrieve the respective attribute for permission decision. Both Hyperledger Fabric supporting databases

(CouchDB and Level DB) are key-value stores, and I have used the ID field as keys.

S_nA is the set of attributes associated with the $subject_n$ and S_nID defined as a key for storing the correlated attributes. Similarly, O_nA is the set of attributes associated with the $object_n$ and O_nID defined as a key for storing the correlated attributes.

P_nSA and P_nOA are the sets of subjects and Objects determinative attributes in the policy of n. Same as attributes, Policy ID (P_nID) is a required field to store attributes. For every policy, the determinative attributes (P_nSA, P_nOA) have been stored along with the policy's rules.

$S_nA = \{S_nID, \{\{S_nA_1, value\}, ..., \{S_nA_n, value\}\}\}$

$O_nA = \{O_nID, \{\{O_nA_1, value\}, ..., \{O_nA_n, value\}\}\}$

Defining SA as the set of all subjects attributes and OA as the set of all objects attributes, we have:

$SA = (S_1A \cup S_2A \cup ...S_nA)$

$OA = (O_1A \cup O_2A \cup ...O_nA)$

$P_nSA \subseteq SA$

$P_nOA \subseteq OA$

$P_nSA, P_nOA \in P$

$P_n = \{P_nID, P_nSA, P_nOA, rules\}$

Resulting, we have the following attributes for the subject S_1A and object O_1A:

$S_1A = \{\text{“}s001\text{”}, \{\text{“}status\text{”}, true\}, \{\text{“}expiration\text{”}, \text{“}2020-05-12\text{”}\}, \{\text{“}libraryGroup\text{”}, 12\}\}$

$O_1A = \{\text{“}r001\text{”}, \{\text{“}libraryGroup\text{”}, 12\}\}$

The policy P_1 with the id “policy01” is as following:

$P_1 = \{\text{“}policy01\text{”}, S_1A, O_1A, \{\text{“}status == true\text{”} \wedge \text{“}expiration\text{”} > \text{“}1Day\text{”} \wedge \text{“}user.libraryGroup\text{”} == \text{“}resource.libraryGroup\text{”}\}\}$

6.4.1 JSON data format

In my implemented solution, the access control data (attributes and policies) is followed by the JSON format. Using JSON, as a widespread data format, can be used by a broad range of applications. An example of a sample policy, including the subject (user) attributes and object (resource) attributes, is illustrated in the following JSON code snippet. Based on the presented policy, the subject and the resource attributes, the subject has valid access permission to the resource, as the subject has valid Identifier (ID), active status, non-expired membership and the subject library group matches with the resource library group. If one of the subject's attributes does not pass the policy rules, the access permission will be denied.

Listing 6.1: Definition of a sample access control policy, subject and resource

```
policy = {
''policyID'': ''policy01'',
attributes: {
''user'': {
```

```
    ''status'': ''Active'',
    ''expiration'': ''Date of expiration'',
    ''libraryGroup'': ''Group ID''
        },
''resource'': {
    ''libraryGroup'': ''Group ID''
    }
                },
''rules'': {
    ''user.status'': {
    ''comparison_type'': ''boolean'',
    ''comparison'': ''boolAnd'',
    ''value'': true
                },
    ''user.expiration'': {
    ''comparison_type'': ''datetime'',
    ''comparison'': ''isMoreRecentThan'',
    ''value'': ''1DAY''
                },
    ''user.libraryGroup'': {
    ''comparison_target'': ''libraryGroup'',
    ''comparison_type'' ''numeric'',
    ''comparison'': ''isStrictlyEqual'',
    ''field'': ''resource.libraryGroup''
                }
            }
        }
    }
}

subject = {
subjectID: ''s123'',
attributes:{
       ''status'': true,
       ''expiration'': ''2020-05-12'',
       ''libraryGroup'': 12
}
}
resource = {
resourceID: ''r123'',
Attributes:{
    ''libraryGroup'': 12
}
}
```

6.5 System evaluation

In this section, I evaluate system performance. I used Hyperledger Caliper to generate workloads and measure its performance based on customized benchmarks and various configurations. The program codes of the project implementation and evaluation experiments are available in the Github repository [2]

6.5.1 Performance parameters, test environment configuration and assumptions

I have evaluated each component of the system against two different databases, Couchdb and Goleveldb and two orderer services, Raft and Kafka. I also present the results of the evaluation based on the Solo orderer.

Raft [3] is a Crash Fault-Tolerant (CFT) ordering service based on the implementation of Raft protocol. Raft simply follows the "leader and follower" model. The leader makes decisions, and the followers follow the leader. The leader changes frequently, and every follower can be a candidate to become the next round leader.

Similar to Raft, Kafka [4] is a CFT ordering service and follows the "leader and follower" model. However, it uses ZooKeeper [5] to manage clusters. Zookeeper keeps track of the status of the Kafka cluster nodes and partitions. Solo is a single ordering node, and it is not fault-tolerant. It is developed to be used only for testing purposes.

I have tested the application on a Virtual Machine (VM) running Google Cloud Platform, VM to collect performance analysis data. The machine type is n2-highcpu-8, which includes eight virtual CPUs and 8 GB of memory. All the tests are run on the same virtual machine, as Caliper emulates workload distribution between several clients.

The default number of blockchain clients is 10 (each client emulated by a different NodeJS[6] process), the default number of transactions is 5,000, and the default transaction type is policy decision transaction, which queries the related data from ledger based on access request and determines the result of the access request. The default database for Raft and Kafka is GoLevelDB. The default number of organizations is two, and the default number of peers is one.

6.5.2 Network topology

Hyperledger Fabric offers a distributed infrastructure that allows multiple organizations to interact. The concept of channels is introduced to separate the communication between organizations and provide private data. Each organization can join multiple channels and establish private communication resulting in a separate ledger for each channel. Each organization can contribute multiple peers, and these peers are

[2] https://github.com/hyperledger-labs/hyperledger-fabric-based-access-control
[3] Raft. https://raft.github.io/
[4] Apache kafka. https://kafka.apache.org/
[5] https://zookeeper.apache.org/
[6] https://nodejs.org/en/

responsible for maintaining a copy of the ledger and executing smart contracts. Smart contracts are installed on peers and then defined on the respective channel. Ordering service consists of multiple nodes called orderer responsible for reaching deterministic consensus (unlike public platforms such as Ethereum and Bitcoin with the possibility of a fork) and ordering the transactions and bundle them into blocks.

Figure 6.6 illustrates an example of a Fabric network. Assume we have three organizations that offer digital library services. Library A and library B are interested in offering an integrated service to their users, and the service is limited to their registered users. The first step is defining a consortium and storing it in the network configuration. Channel 1 will then be created based on network configuration. Each organization (library) contributes one peer to the channel. In the next step, chaincodes responsible for implementing the ABAC system will be installed on both peers and defined on Channel 1. The data stored on the ledger will be only accessible to the members of Channel 1. It is also possible to restrict the users' access level through Fabric membership service and implement more granular access at the Fabric network level. So far, this reflects the network topology for the implemented application. It includes two organizations (org1 and org2) connecting through Channel 1, two peers (peer0.org1 and peer0.org2). I have tested the application through two different endorsing services. Raft ordering service established through three orderer nodes (ordered 0, orderer 1, orderer 2). Raft is a CFT that can withstand one failure out of three nodes. Kafka is established through two orderer nodes (orderer 0 and orderer 1).

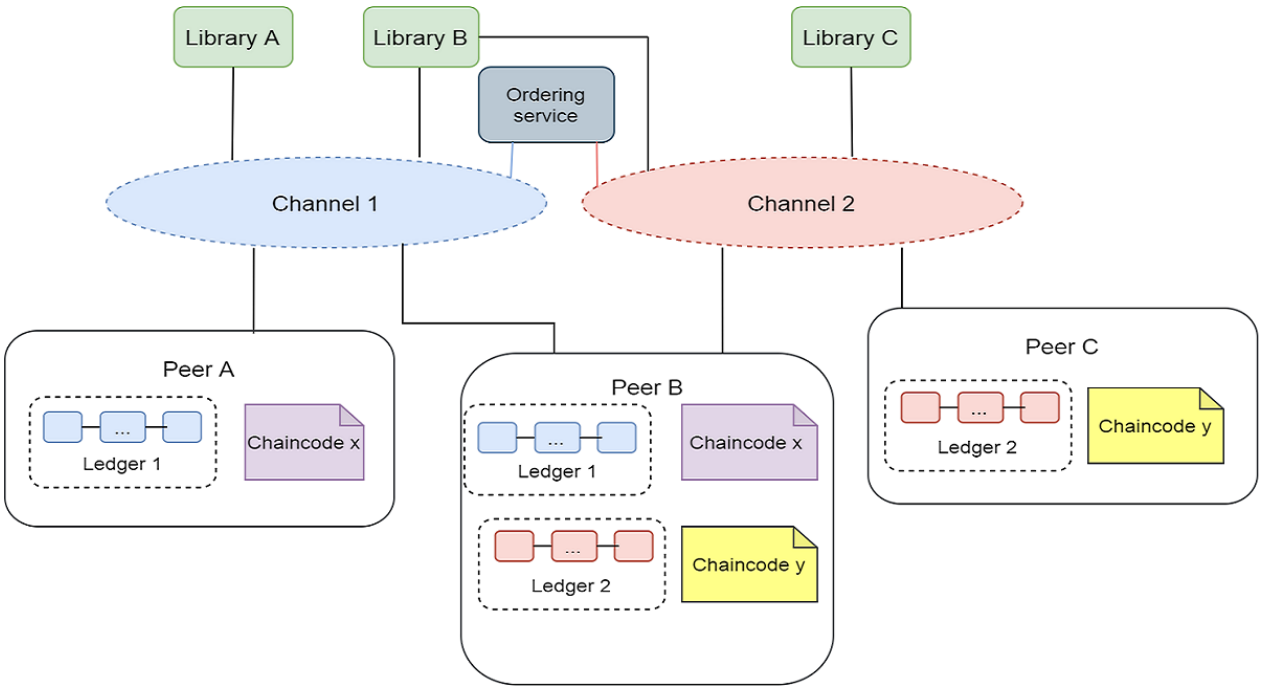

Figure 6.6: Network topology.

As an extension to this scenario, consider that Library B is interested in integrating another service with a new organization (library C). To implement the ABAC access control, they are required to store the related attributes and policies on the ledger. They can establish a new channel (Channel 2) and maintain a separated

ledger on their peers. The data stored on the new ledger is not exposed to the organizations joined Channel 1. Library B either can use the same peer (Peer B) for both channels, or it can provide two separate peers to contribute separately in Channel 1 and Channel 2.

6.5.3 Performance evaluation results

Figure 6.7 shows the average latency (in seconds) for three different transactions, Record attributes, PDP, and query data from the ledger based on Kafka orderer. Record includes storing both subjects and resources attributes and storing policies; there are three transactions, record objects' attributes, record subjects' attributes and record policies. Since these three transactions basically do the same task, their extracted results were very similar. I have considered the average latency of these three transactions for Record. Query queries the stored data from the ledger, including attributes and policies. PDP queries the related data from ledger based on access request and determines the result of the access request.

In every round of the test, we configured the test with a different number of transactions. Although the average latency increases with the number of transactions, the increase is not sharp, growing very slowly. As illustrated in the following results, the system can process a throughput of around 270 transactions per second (PDP decisions) with an average latency of 0.54 seconds.

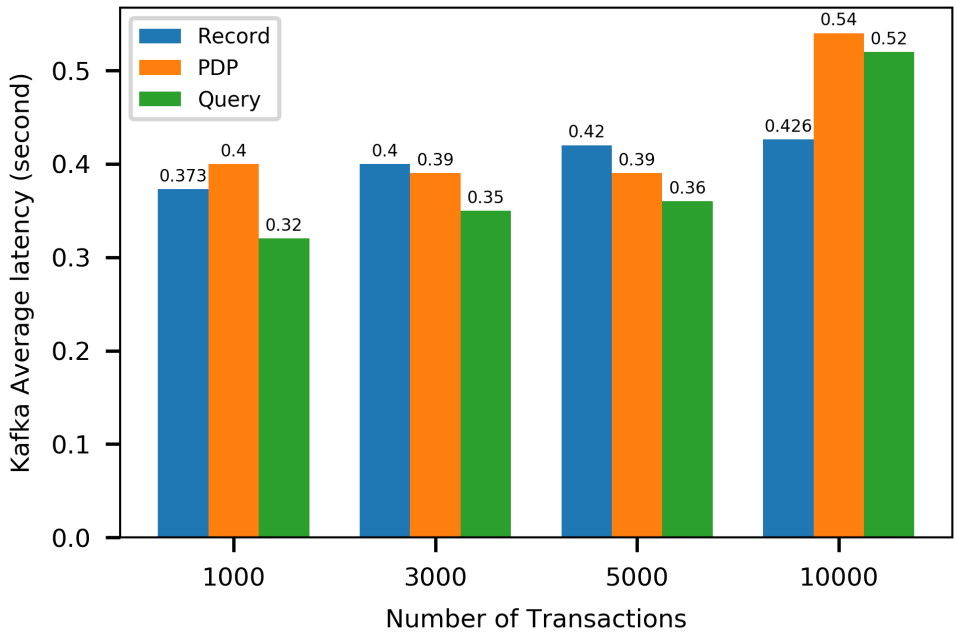

Figure 6.7: Average latency of Kafka as a function of the type of transactions.

Figure 6.8 shows the average latency (in seconds) for the same three different transactions, based on the Raft orderer. The test result is based on a different number of transactions. Similarly, the average latency increases with the increase in the number of transactions. The test run failed for the Raft orderer with

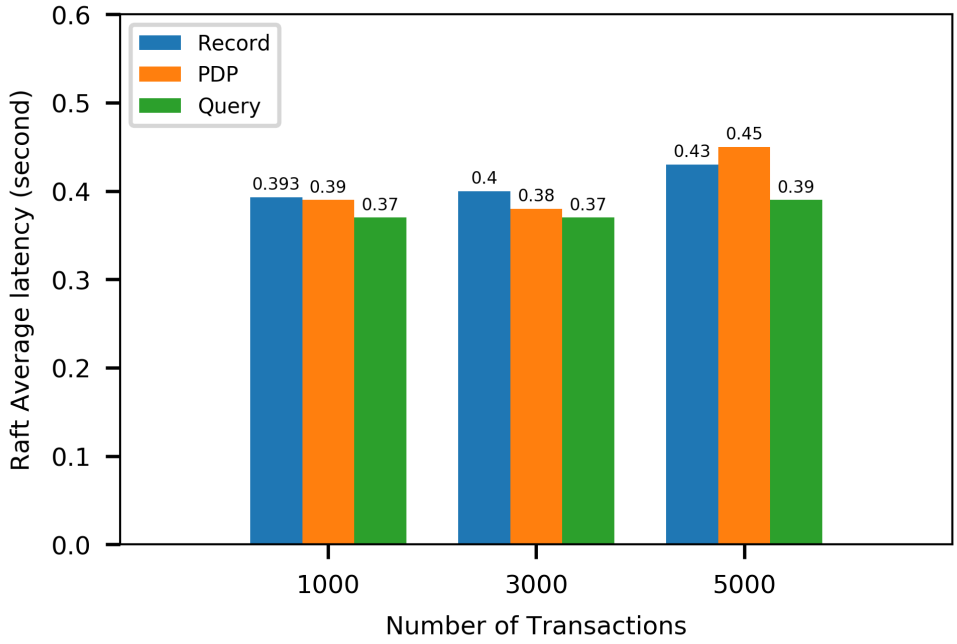

Figure 6.8: Average latency of Raft under different transactions.

10,000 transactions due to VM Memory limitation. Later, the resource consumption result indicates that Raft consumes almost 4.5 times more memory compared with Kafka, which explains why the test failed in the middle for 10,000 transactions with Raft orderer.

For both Raft and Kafka orderers, increasing the number of transactions increases the average latency for the record attributes transaction. Policy decision transaction has the minimum average latency for both Kafka and Raft, based on 3,000 transactions. For Record attribute and query data transactions, the average latency in Raft is slightly higher than Kafka. For the policy decision transaction, the average latency in Raft for 1,000 and 3,000 transactions is slightly lower than Kafka, but for 5000 transactions, the result is the opposite.

Figures 6.9 and 6.10 show the average latency and throughput for the policy decision transaction for Raft and Kafka based on different send rates. The number of transactions for these two tests is 5,000. The dendrite of 200 Transactions Per Second (tps) is an optimal point for Kafka as it exhibits the lowest average latency, and the average latency increases sharply after the send rate of 200 tps. Raft is also exhibiting an increase in average latency after the send rate of 200 tps, but comparatively performing better than Kafka. The reason could be related to the fact that Raft is has used three orderer nodes to handle the ordering service, and Kafka is using two orderer nodes as it is shown in table 6.1.

The maximum throughput for both Raft and Kafka orderers is at the send rate point of 300 tps; afterwards, the throughput drops for both of them. Overall, in terms of throughput and average latency, Raft performed better than Kafka.

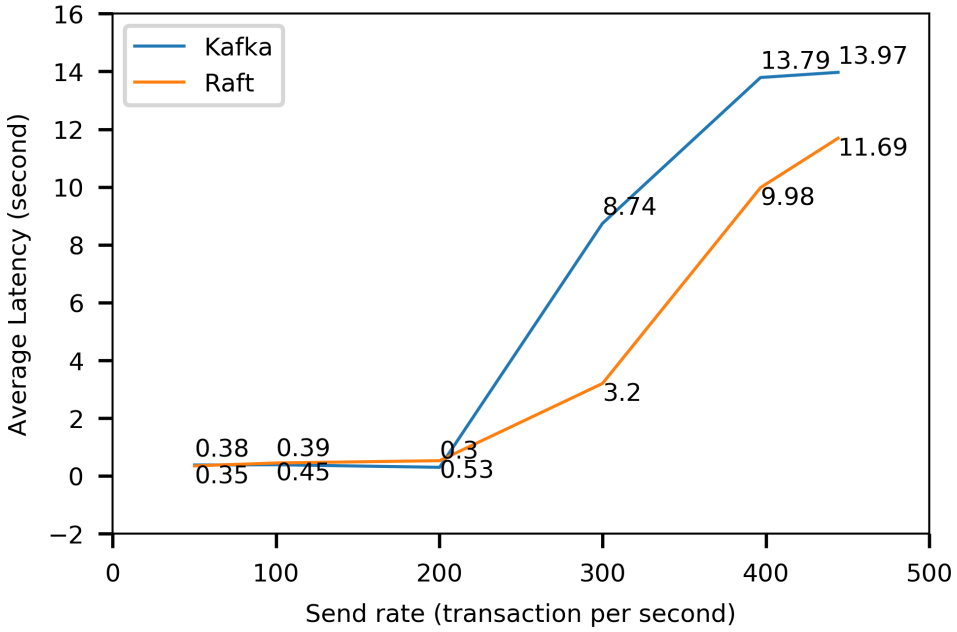

Figure 6.9: Average latency of Raft and Kafka based on different transactions.

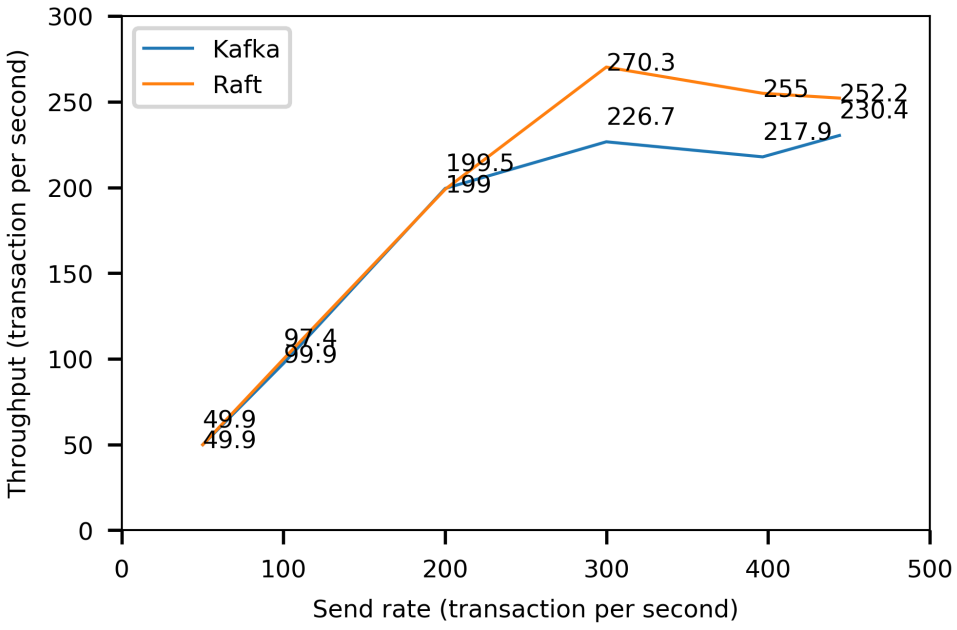

Figure 6.10: Throughput of Raft and Kafka under different transactions.

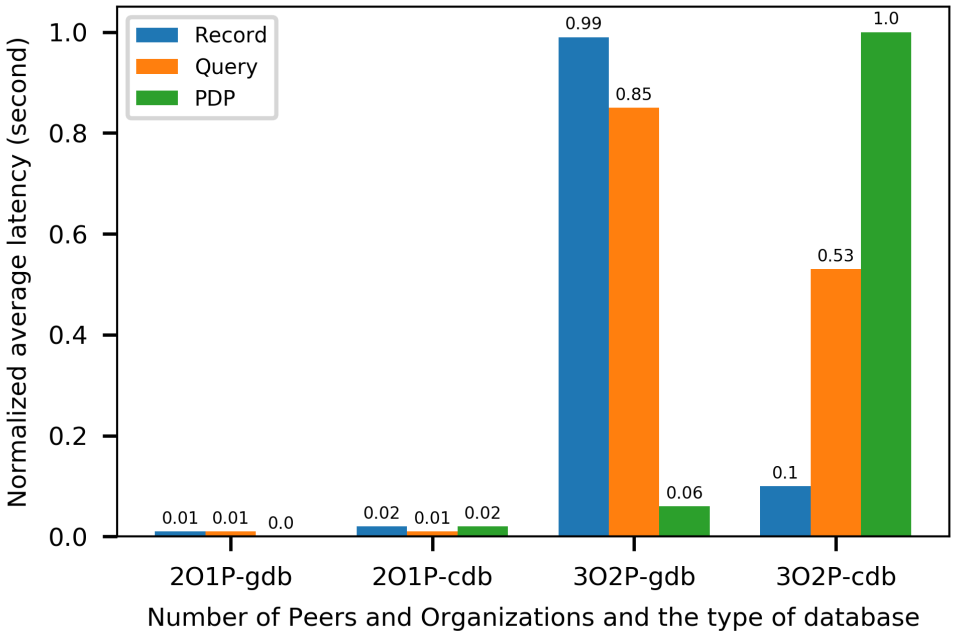

Figure 6.11: Throughput of Raft and Kafka under different transactions.

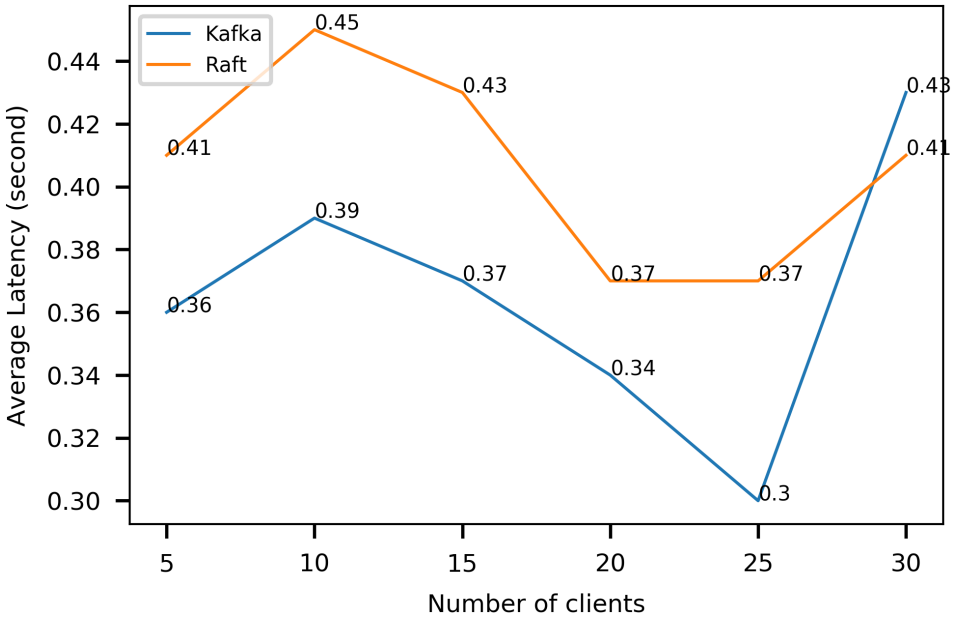

Figure 6.12: Average latency based of Raft and Kafka with different number of clients.

Table 6.1: Resource consumption for Raft and Kafka.

Orderer	Name	Memory(max)	Memory(avg)	CPU(max)	CPU(avg)	Traffic In	Traffic Out	Disc Read	Disc Write
Raft	dev-peer0.org1	74.4MB	71.MB	32.25%	28.21%	13.8MB	5.3MB	0B	0B
	dev-peer0.org2	73.3MB	69.7MB	33.23%	28.28%	13.8MB	5.3MB	0B	0B
	peer0.org1	379.3MB	369.4MB	67.21%	56.32%	31.1MB	23.7MB	0B	21.8MB
	peer0.org2	284.0MB	274,1MB	70.78%	55.66%	31.1MB	23.6MB	4.0KM	21.8MB
	orderer1	554.1MB	535.4MB	26.11%	15.41%	22.4MB	59.1MB	0B	37.2MB
	orderer2	525.3MB	506.5MB	18.78%	11.88%	27.6MB	28.7MB	0B	37.0MB
	orderer0	513.3MB	494.7MB	19.40%	11.63%	27.5MB	10.6MB	0B	37.2MB
Kafka	dev-peer0.org1	73.5MB	72.8MB	17.87%	15.26%	7.5MB	2.5MB	0B	0B
	dev-peer0.org2	64.5MB	62.8MB	18.73%	15.69%	7.5MB	2.5MB	0B	0B
	peer0.org1	295.9MB	286.2MB	52.08%	49.15%	27.1MB	17.0MB	368.0KB	21.2MB
	peer0.org2	294.1MB	282.7MB	51.54%	48.38%	27.1MB	17.1MB	152.0KB	21.2MB
	orderer0	121.1MB	113.1MB	25.80%	23.30%	29.6MB	11.1MB	4.0KB	18.4MB
	orderer1	124.1MB	115.2MB	21.87%	20.25%	29.7MB	46.5MB	276.0KB	18.4MB

Figure 6.11 shows the effect of increasing the number of organizations and peers and comparing two different databases, GoLevelDB and CouchDB. Increasing the number of organizations and peers increases the average latency for all three transactions. For the CouchDB database with three organizations and two peers, the average latency increases sharply to 43.81 seconds for the policy decision transaction, which is 17.73 times more than GolevelDB and 64.42 times more than the same database, with two organizations and one peer. It shows that CouchDB does not perform acceptably in terms of scalability for the presented access control application.

Figure 6.12 shows the average latency for both Raft and Kafka based on the different number of clients. This test is run based on 5,000 transactions and the policy decision transaction. For Kafka, 25 number of clients is like an optimal point that has the lowest average latency (0.3 seconds). However, 25 is an optimal point for the system with current resources. As the graph shows, for Raft, there are two points for the minimum average latency, corresponding to 20 and 25 clients. In general, Kafka performs better under 25 clients, but after 25 clients, it shows a sharp increase in the average latency. Although it shows that the system is not scalable after 25 clients, it significantly depends on the computation power and limitations of the VM instance. We have repeated the test with a more powerful VM instance, and the average latency was 0.36 second for Raft and 0.31 second for Kafka with 60 clients. Similar to the previous experiments (figures 6.9, and 6.10), Kafka performs better under the optimal point, but when the number of clients is increased to more than 30 clients, Raft performs better.

Table 6.1 presents the resource consumption for policy decision transactions based on 5000 transactions for Kafka and Raft. As the presented numbers in the table demonstrate, Raft consumes 4.33 times more memory on average in comparison with Kafka. It clarifies that the early test results with 10,000 transactions with Raft ordered failed because the VM ran out of memory.

6.5.4 Security considerations

The presented system alleviates security and privacy concerns that centralized access control systems suffer by offering a pluggable distributed application for access control that provides reliable auditing data for accountability and non-repudiation. However, there are additional precautions that need to be taken. It allows engineers to reuse the presented access control component, which is decoupled from the underlying resources that are protected. While this provides ease of implementation, new concerns arise. We need to ensure that there are no other ways to access protected resources. It is also essential to ensure that the local world state database is not tampered with or accessed by the system administrators. The possible solution includes enforcing only one access control channel (the blockchain) and secure, tamper-proof storage for logs (for example, another blockchain) [17]. Although using blockchain to store resources increases its resiliency, there are data integrity vulnerabilities. Further research is needed to assess the practical implications of such an approach.

6.6 Chapter summary

Reliable accountability mechanisms are essential for audits. In this paper, I discussed how permissioned blockchains could be helpful as a trustable backend in access control systems by providing a solid basis for audits. I proposed a distributed ABAC system based on Hyperledger Fabric, focusing on audibility and scalability. I validated the solution through a decentralized access control management application in digital libraries. First, I presented a comprehensive review of studies focusing on blockchain-based access control studies. Then I presented the system architecture and implementation details, where I implemented the PDP, PAP, and AM components using smart contracts on-chain, and the PEP was implemented off-chain - based on the blockchain clients' requirements.

I considered various experimental evaluation parameters based on the Hyperledger Caliper framework in terms of system performance. The evaluation results indicate that the presented system effectively handles a throughput of 270 transactions per second, with an average latency of 0.54 seconds per transaction.

6.7 Contribution of the work presented in this chapter

This chapter presented a decentralized attribute-based access control system based on Hyperledger Fabric. All ABAC components in the context of policy-based architecture were implemented as smart contracts and automatically enforced access control decisions. The system was evaluated through a case study of digital libraries. A complete performance analysis based on two different orderer services, Raft and Kafka, two databases, CouchDB and GoLevelDB, and various network configurations, indicated the system's acceptable performance and scalability.

This study is the result of my internship project with the Linux Foundation, Hyperledger project. The contribution of this chapter has been published in the following paper:

Rouhani, Sara , Rafael Belchior, Rui S. Cruz, and Ralph Deters. "Distributed attribute-based access control system using a permissioned blockchain." world wide web(2021), doi: 10.1007/s11280-021-00874-7.

7 Data trust framework

Data sharing has become a big concern regarding privacy and confidential issues, abusing data, and legal and ethical violations. The lack of a transparent and trustworthy framework for data trust hinders many data owners from sharing their data, which could be vital for many research purposes. Data sharing is not merely a big concern for data owners, but also data users are concerned about the trustworthiness and reliability of the provided data at the origin. Hence, trust is a two-way problem for both data owners and data users.

Data trust is a fairly new concept that aims to facilitate data sharing by forcing data users to be transparent about the process of sharing and reusing data. Data trust entails legal, ethical, governance and organizational structure as well as technical requirements for enabling data sharing. Previous studies have suggested the potential of web observatory [124] and institutional repositories [9] for implementing data trust.

Although many other studies investigate blockchain potential for data sharing and access control, they are mostly scattered studies focused on a particular step or specific aspect of data sharing or have taken side one of the parties in data sharing addressing only data owners concerns.

This study proposes an end-to-end framework for data trust based on blockchain, which ensures the trustworthiness and quality of the data at origin for data users and ethical and secure usage of data for data owners.

I introduce a trust model to assess input datasets' trustworthiness using three parameters: data owner endorsement and reputation, data asset endorsement and data owner confidence level in the provided dataset. I have recorded all these parameters on the ledger using implemented smart contracts, and they will be updated with every new transaction.

Adaptive transaction validation is applied using Hyperledger Fabric state-based endorsement based on datasets trust value. I have conducted a comprehensive performance analysis to demonstrate the system's efficacy in handling large sets of transactions and scaling across multiple organizations. The proposed system presents all the properties required for data trust. At the same time, it benefits from transparency, immutability, security offered by blockchain technology, and smart contracts' automation capabilities.

This chapter addresses the following questions:

- Why blockchain is a suitable infrastructure for implementing data trust?
- How can we implement data trust's required properties using blockchain?
- How can we develop an end-to-end solution for data trust using blockchain?

7.1 Data trust studies

Various studies have investigated blockchain potential for trusted data sharing. Some studies have considered incentive mechanisms to encourage data owners to share their data without losing control and ownership. I have assessed input datasets' trustworthiness through multiple trust models such as reputation-based models, smart contract verification, and algorithmic solutions.

Hawlitschek [73] et al. presented a literature review on trust in the context of blockchain technology and sharing economy. Karafiloski et al. [87] discuss the blockchain-based solutions associated with trust, data ownership, protection and security in big data.

Table 7.1: Blockchain based data sharing and quality control studies.

Study	Incentive-based	Quality control	Blockchain network	Implementation	Application domain
Xuan et al. [187]	Yes	-	-	Partially	General data sharing
Shrestha et al. [163]	Yes	-	Ethereum (public)	Yes	General data sharing
Shen et al. [161]	Yes	-	-	Partially	Cloud data
Chen et al. [39]	Yes	Yes	Ethereum (public)	Yes	Internet of Vehicles
Zhu et al. [205]	Yes	-	-	Partially	Medical data
Su et al. [166]	Yes	Yes	-	Yes	Disaster rescue
Casado et al. [33]	-	Yes	-	No	IoT
Zheng et al. [204]	-	Yes	Ethereum (public)	Yes	Medical data
Cappiello et al. [32]	-	Yes	Ethereum (public)	Yes	General data sharing
Huang et al.[81]	-	Yes	Ethereum (public)	Yes	Crowd sensing
An et al. [11]	-	Yes	-	Yes	Crowd sensing
Wang et al. [174]	Yes	Yes	Gervais's Bitcoin (Simulator)	Yes	Crowd sensing
Zovolokina et al. [195]	Yes	Yes	Hyperledger Fabric (permissioned)	Yes	Digital car dossier
Dedeoglu et al. [51]	-	Yes	Custom (private)	Yes	IoT
Shala et al. [159]	Yes	-	-	Yes	IoT
Yue et al. [193]	-	-	-	-	Big data sharing
Brandao et al. [24]	-	-	-	-	Smart places
Kang et al. [86]	-	Yes	Consortium blockchain	Yes	Vehicular edge computing
Yang et al. [189]	-	Yes	-	Yes	Vehicular networks
Kim et al. [89]	-	-	-	Yes	Wireless sensor network
Kochovski et al. [90]	-	-	Ethereum (public)	Yes	Fog computing
Wei et al. [179]	-	-	-	Yes	Cloud data
Choudhury et al. [43]	-	Yes	Hyperledger Fabric (permissioned)	Yes	Medical data
Hang et al. [72]	-	Yes	Hyperledger Fabric (permissioned)	Yes	Fish farm
Presented system	-	Yes	Hyperledger Fabric (permissioned)	Yes	General data sharing

Table 7.1 provides the summary of blockchain-based data sharing, data trust and quality control studies.

Shala et al. [159] introduce an incentive mechanism to motivate low trusted peers in the IoT network to increase their trust score. The incentivization system uses control loops that contain a target trust score. For the service providers with low trust scores, a package of incentives, such as discounts for other services, will be sent to encourage them to offer a better service in exchange for the promised benefits. In [205], authors present an incentive-based model to encourage medical data owners to share their high-quality data (real and practical) and earn revenues, as well as miners who get benefit by participation and validating transactions.

Wang et al. [174] introduce a privacy-preserving incentive mechanism to achieve high-quality contribution in crowdsensing. The trust model motivates participants to share their high-quality sensing data and profit in the form of Bitcoin or Monero cryptocurrencies. Miners verify the quality of data and earn revenue as well. Zavolokina et al. [195] provides a financial incentive for participating in the network and provides high-quality data for car dossiers. The system expects to fix errors by punishing harmful behaviour. They employ smart contracts for automatically calculating and enforcing incentives. [163] utilizes blockchain and smart contracts to encourage data owners to share their research data without losing control and ownership of it. The system incentivizes the participants by providing access to aggregate, anonymized data in exchange for involving as miners in the network.

Kang et al. [86] propose a subjective logic model to assess nodes' reputation to ensure high-quality data sharing in the vehicular network. Dedeoglu et al. [51] present a trust model to assess the trustworthiness of data observed by sensor nodes in the IoT network. The model consists of three elements: evidence from other neighbour sensor nodes observations, the data source's confidence, and its reputation. They also employ blockchain to control shared data trustworthiness by detecting inaccurate or suspicious data captured by IoT devices or mobile crowdsensing. Choudhury et al. [43] ensure data trustworthiness while maintaining data privacy. Regulatory agencies assess the quality of data as network participants. Data privacy is ensured by creating activity-specific private channels. An et al. [11] present a lightweight consensus mechanism called delegated proof of reputation (DPoR) to solve the heavy computation problem appropriate for data quality control in crowdsensing nodes. Huang et al. [81] examine the quality of collected data from sensor nodes in the crowdsensing network through verification rules embedded in smart contracts. Su et al. [166] have designed a two-tier reinforcement learning (RL)-based incentive algorithm to improve high-quality data sharing. Casado et al. [33] also present a cooperative algorithm based on game theory in the edge computing layer to promote data quality and false data detection. Table 7.1 outlines the summary of incentive-based and quality control for data stewardship using blockchain technology.

7.2 System architecture

As I discussed earlier, the proposed system aims to establish a data trust framework beneficial for both data owners and data users. To tackle this goal, I present two main components in the system architecture. 1) a trust model to examine the trustworthiness of input datasets and 2) a secure and traceable access control management. Figure 7.1 presents the proposed data trust framework architecture.

I model trust for the input datasets as follows. For any initial dataset, the system calculates its trust value through a blockchain-based application. This value is used to ensure only trusted datasets are confirmed, and the system only records trusted data assets on the ledger. Section 7.3 explains which parameters are involved and how I have calculated the trust value.

The datasets with lower trust values are considered suspicious, and they are required to be validated by

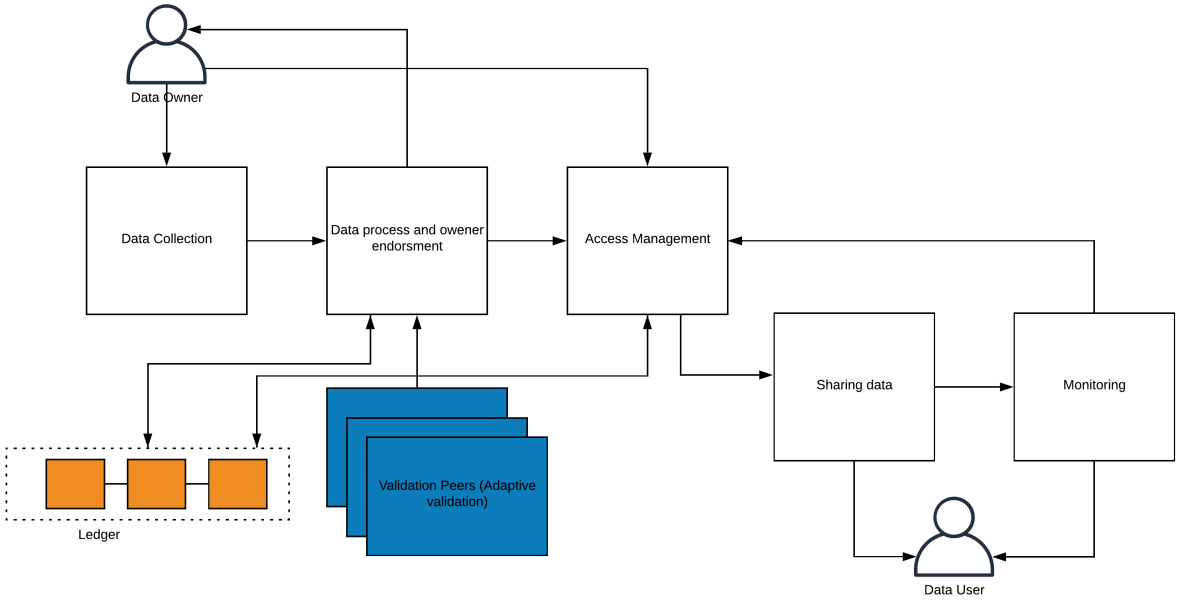

Figure 7.1: End to end data trust architecture with adaptive validation.

more verifiers. This adaptive selection of the number of verifiers provides an acceptable trade-off between the system's data assets trustworthiness and its performance and resource consumption. In order to prevent a data breach by data investigators, they would have access to a small chunk of data. The accuracy and quality of the data are examined through that small chunk.

Once the datasets' information is recorded as data assets on the ledger, the data users interested in accessing a dataset can prepare a request to access the dataset. The data owner will receive the requests directly, and they will decide to give access to their datasets under which terms and conditions. Using blockchain and smart contracts, all transactions are automatically enforced, and there are no third parties involved. The access control policies could be served through a smart contract, created and stored on the blockchain directly by the resource owner. Moreover, the data owners can transparently query the immutable and permanent storage of blockchain and trace who had access to their dataset previously and who currently have plus all the history of access requests despite the data owners' response type. Section 7.4 describes the details of implemented smart contracts for access control and consent management.

7.3 Trust model

This section discusses how I calculate the trust value for the input datasets. Later, this value will be available for the data users interested in a particular dataset. Moreover, the system adaptively includes this value to define the number of verifiers for confirming the dataset. The higher trust value requires fewer number of verifiers; therefore, the dataset will be verified faster.

7.3.1 Terminology

I call datasets the data assets as they are assets used in the distributed data trust framework to manage them and control access to them. Every data asset has an owner (data owner) that has full control over the data. Every data asset has a unique identifier (key).

Data owners are the one who provides the data asset. The data owner has full control to decide who will have access to the data, for what purpose and the access permission level and conditions. The system provides a transparent history of all granting and revoking access permissions executed by the data owner.

A data subject refers to any person whose personal data is being collected and can be identified, directly or indirectly, via an identifier such as a name, an ID number, address, or his or her information on social media.

The datasets provided by data owners may or may not include personal data. In any case, they require a precise procedure to share datasets. Sharing datasets that include personal data requires additional mechanisms to protect data subjects by preparing the dataset in a way that does not contain individually identifiable information any more. The pre-requisite steps must be provided to ensure sufficient privacy safeguards are in place for protecting data subjects' personal information. These mechanisms could include de-identification or anonymization. De-identification refers to erasing or replacing personal identifiers to make it difficult to re-establish a link between the individual and his or her personal information. Anonymization refers to the permanent removal of the link between the individual and its personal data to the degree that it would be practically impossible to re-establish the link [95].

Data users are potential users interested in datasets, such as data analysts, data miners, or data scientists to extract knowledge, insights, and meaningful or profitable patterns from the dataset. They use the proposed data trust framework to discover and request access to their intended datasets. They also require trust in the quality of the provided dataset.

7.3.2 Input datasets trust assessment model

I have used a trust factor in the represented data trust framework to examine the level of trust in the input datasets. I have considered this trust factor to determine the number of verifiers to examine the dataset's trustworthiness before confirming the dataset and record it as a valid data asset in the ledger.

In the adaptive trust model, trust is modelled for the data asset with the key of i as:

$$Trust_i = f(reputation_{uj}, endorsment_{uj}, \lambda(confidence_i)) \quad \lambda \in [0,1] \tag{7.1}$$

where f is a function with three input parameters, including data owner's reputation, $reputation_{uj}$, data owner's endorsement, $endorsment_{uj}$, and data owner's confidence for the provided dataset, $confidence_i$. The λ factor determines the impact of the selected confidence in the total trust score. Later, I will discuss

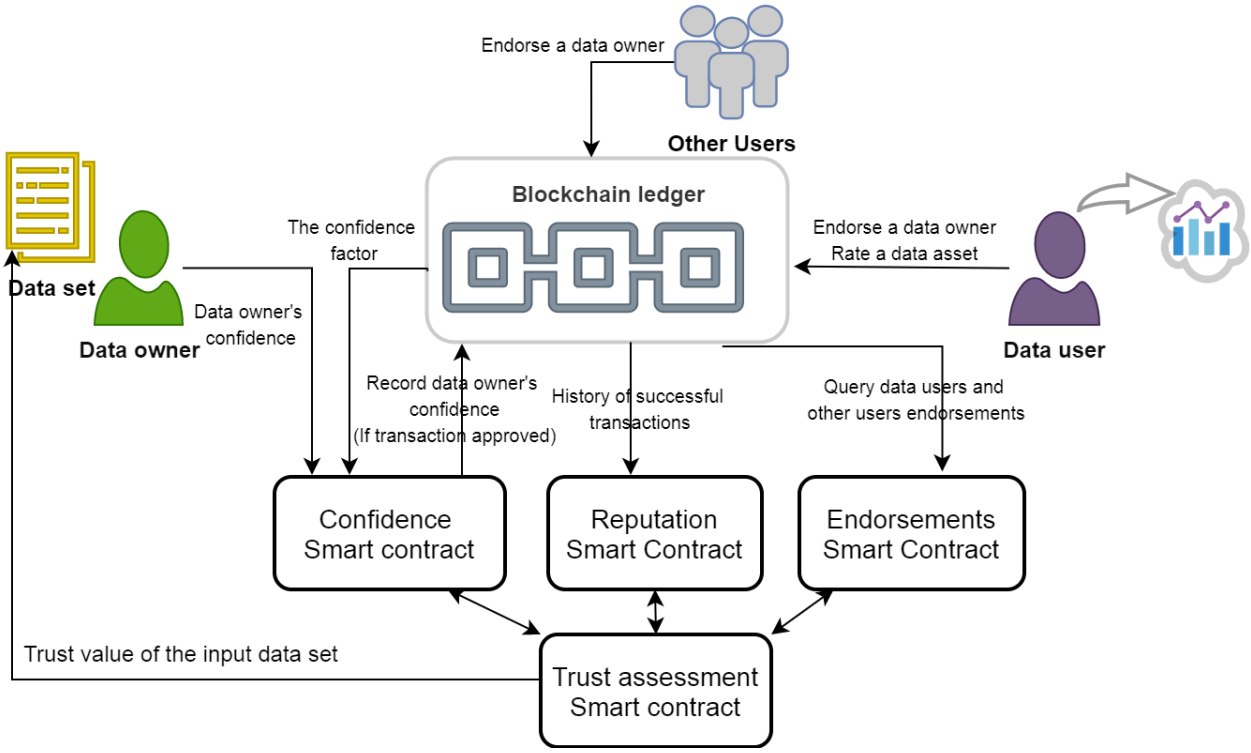

Figure 7.2: Trust assessment of input datasets.

how the λ coefficient is calculated based on the received assessments from data users. Figure 7.2 represents trust assessment model for input datasets.

I have implemented three smart contracts for calculating reputation, endorsement, and confidence parameters. The smart contracts read the required values from the ledger and calculate the specified parameter value. Subsequently, the smart contract stores the result of each parameter on the ledger. Those results will be included in the next round of the parameter calculation, such as calculating the confidence parameter and calculating the total trust value, which will be used for system analysis in the future. Finally, I have implemented another smart contract to invoke the three input parameters smart contracts and calculate the current dataset's trust value.

Reputation

I calculate the data owner's reputation considering the user's previous successful transactions as well as the minimum and the maximum number of successful transactions for all users, using the min-max normalization:

$$reputation_{uj} = \frac{\sum_{i=1}^{n} T_s \; - \; \min\limits_{\forall\, u \,\in\, U} \sum_{i=1}^{l} T_s}{\max\limits_{\forall\, u \,\in\, U} \sum_{i=1}^{h} T_s \; - \; \min\limits_{\forall\, u \,\in\, U} \sum_{i=1}^{l} T_s} \tag{7.2}$$

where $\sum_{i=1}^{n} T_s$ is the number of successful transactions that the user uj have had so far. Also $\min\limits_{\forall\, u \,\in\, U} \sum_{i=1}^{l} T_s$ and $\max\limits_{\forall\, u \,\in\, U} \sum_{i=1}^{h} T_s$ are respectively the minimum and maximum number of successful transactions for all

users in the framework.

More successful transactions, which means more submitted valid data assets by the data owner, increases the data owner's reputation. Multiple validators examine every input data asset; the number of these validators will be defined based on the trust factor. Because the new users do not have any history of valid data assets, their reputation score is relatively low. As they interact more honestly, they gain more reputation. Subsequently, their following transactions will require fewer validators, and their transactions will get validated faster.

Endorsement

Data owners can receive two types of endorsements. In the first type, any user in the system knows the data owner can endorse one; for example, they could have worked together previously. The second type of endorsement is received from the data users who have previously studied a dataset provided by the current data owner. The data owner receives an endorsement based on the data user's experience regarding the high quality of the data set. The second type of endorsement has a more substantial influence on the data owner's total endorsement score. Endorsement value for the data owner j is calculated as:

$$endorsment_{uj} = 1 - e^{-(\sum endors_{uj} + \alpha \sum endors_{d_{uj}}) \,/\, \beta} \tag{7.3}$$

Where $\sum endors_{uj}$ is the total endorsements that the user uj has received from other ordinary users, and $\sum endors_{d_{uj}}$ is the number of endorsements, the submitted data asset has received in the framework. Also, β is a factor that defines the speed of reaching the highest possible endorsement value, which is one here. α is a positive weight that defines the importance of data asset endorsement versus user endorsement. Figure 7.3 illustrates the effects of alpha and beta on the user total endorsement value assuming $\sum endors_uj$ is 50 and $\sum endors_{d_{uj}}$ is 30.

Confidence

For any initial data set, the data owner enters a confidence value between 0 to 1 ($consideence_i \in (0, 1]$) to express her or his confidence in the provided data set. This value will only be considered in the data asset's total trustworthiness if there is a history of previously entered data sets by the current data owner studied by a data user.

For the new data owners that the system does not have any assessment of their previous confidence value, the initial λ is zero. It means their confidence will not be considered to calculate the trustworthiness of the provided data set. However, this confidence value will be recorded in the system. Later, once a data user rates the data set's quality, the difference between the data owner's provided confidence, and the data user's rate will be calculated as Δ. The proposed calculation of the λ value is given as:

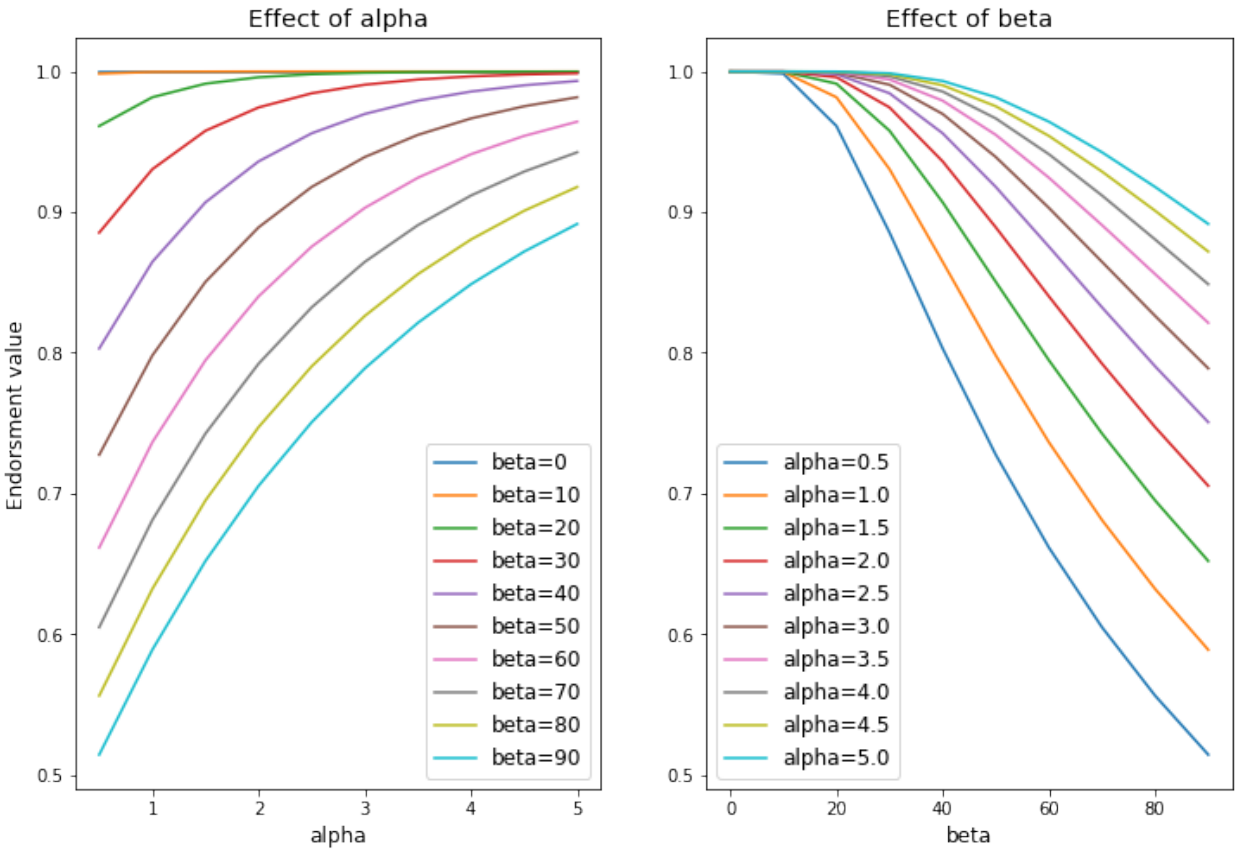

Figure 7.3: The effect of alpha and beta on the data owner's endorsement.

$$\Delta = DataSetRating - DataOwnerConfidence$$

$$\lambda_{new} = \begin{cases} \Delta \geq 0 & min(1, \lambda_{old} + \phi(1 - \Delta)) \\ \Delta < 0 & max(0, \lambda_{old} + \phi\Delta) \end{cases} \tag{7.4}$$

$$\lambda \in [0, 1]$$

$$\phi > 0$$

Where ϕ is a factor that determines the speed of reaching the maximum of λ, which is one. Adaptively, as the data owners provide more accurate predictions about the quality of their data set through their entered confidence value, later their input confidence value will have more effect on the total trust value of the new data set.

7.4 Access management and sharing data assets

As discussed previously, the proposed end-to-end data trust framework addresses the concerns of both data owners and data users. The previous section explained how the proposed data trust model calculates the trustworthiness of input data sets and how it adapts to the confidence of data owners in their provided data sets. This section describes how we can implement a secure and trustable access management system using distributed ledger technology. It also describes how to design smart contracts to meet the requirements of the data trust framework.

Blockchain's features, such as transparency, auditability and trust distribution, along with leveraging smart contracts, make it possible to achieve secure and fine-grained access control, thereby promoting the data-trust framework. This section introduces the access control and consent management components for the presented data trust framework.

Once the data is approved, the data set could either be recorded on a secure cloud storage service provided by the data trust implementing party, or it could stay at the owners' side. In the second case, the owner party is responsible for providing access to the data set, but still, they adopt the blockchain-based access control and consent management system. Despite the location of storing the data sets, for any new data asset added to the data trust framework, a new record will be recorded on the ledger, including data asset id, the owner of the data asset and the hash value of the data asset. Any access permission requires the digital signature of the owner for approval.

Access permission can be granted to a particular user or a sub group of users belong to one or multiple organizations. Three primary smart contracts are responsible for handling access requests, consent management and access provenance. Figure 7.4 illustrates the smart contracts design and interactions.

Access control smart contract receives access requests, checks default access permissions. Suppose the user has access to the data set based on the system's previous rules and prior consent between the data owner

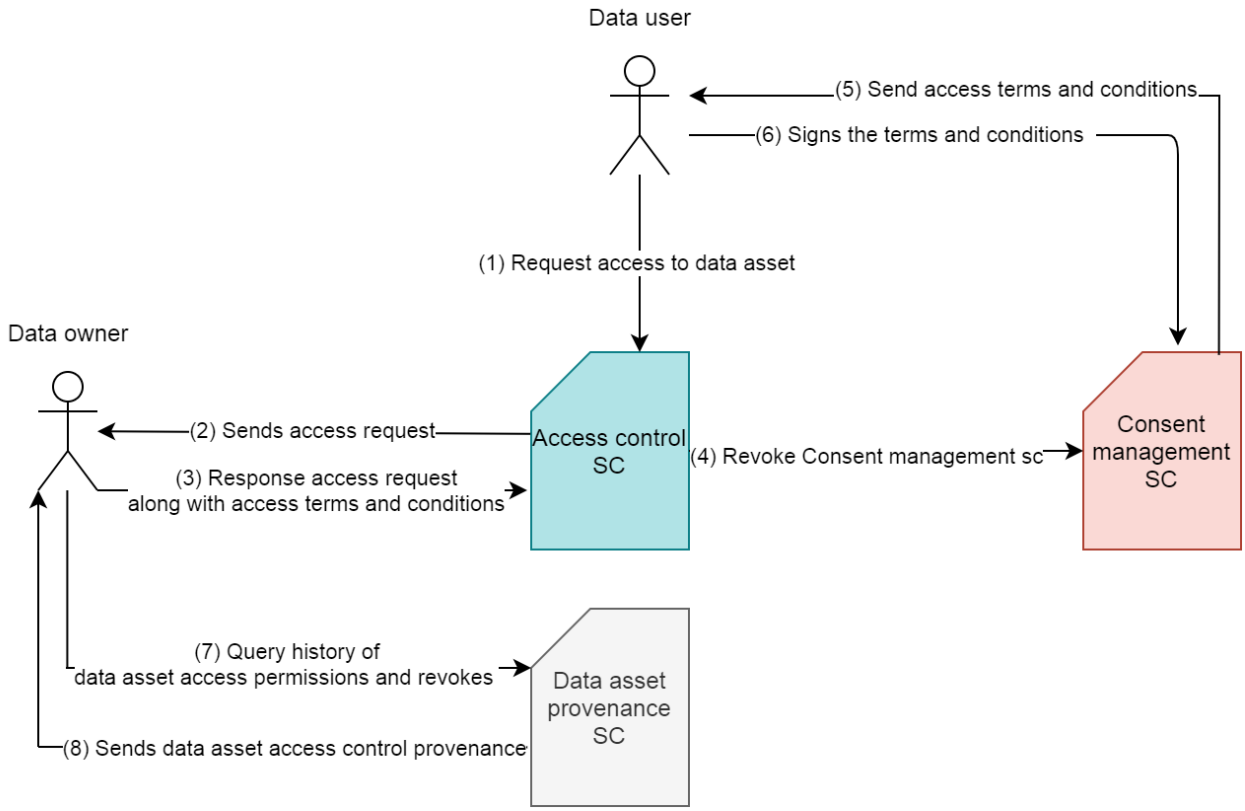

Figure 7.4: Design of smart contracts for access control, consent management and data provenance.

and data user. In that case, it submits a transaction for recording the access request and the result of access permission.

If there are no default access rules that match the current access request, it sends the access request to the data owner. The data owner investigates the access request and decides to accept or reject it based on metadata provided by the data owner. Suppose the response received from the data owner is accepted. In that case, the access control smart contract invokes the consent management smart contract to handle the agreement between the data owner and data user.

The consent management smart contract is responsible for sending the owner's terms and conditions to the data user and collecting the required signatures. It records the agreement to be permanently stored on the ledger.

Data asset provenance smart contract is implemented to query all the access requests, permissions, and revokes toward data assets. All transactions are appended to the blockchain history, whether valid or invalid. Therefore, all access requests are recorded on the ledger despite their acceptance and rejection outcome. They are valuable resources to analyze in the future and detect possible threats. If somebody had tried to compromise the systems' security, all the access attempts were recorded on the transactions' history. Data owners are able to query the data users who currently have access to their data assets as well as query the previous history of access changes. Utilizing smart contracts for querying data asset provenance makes it

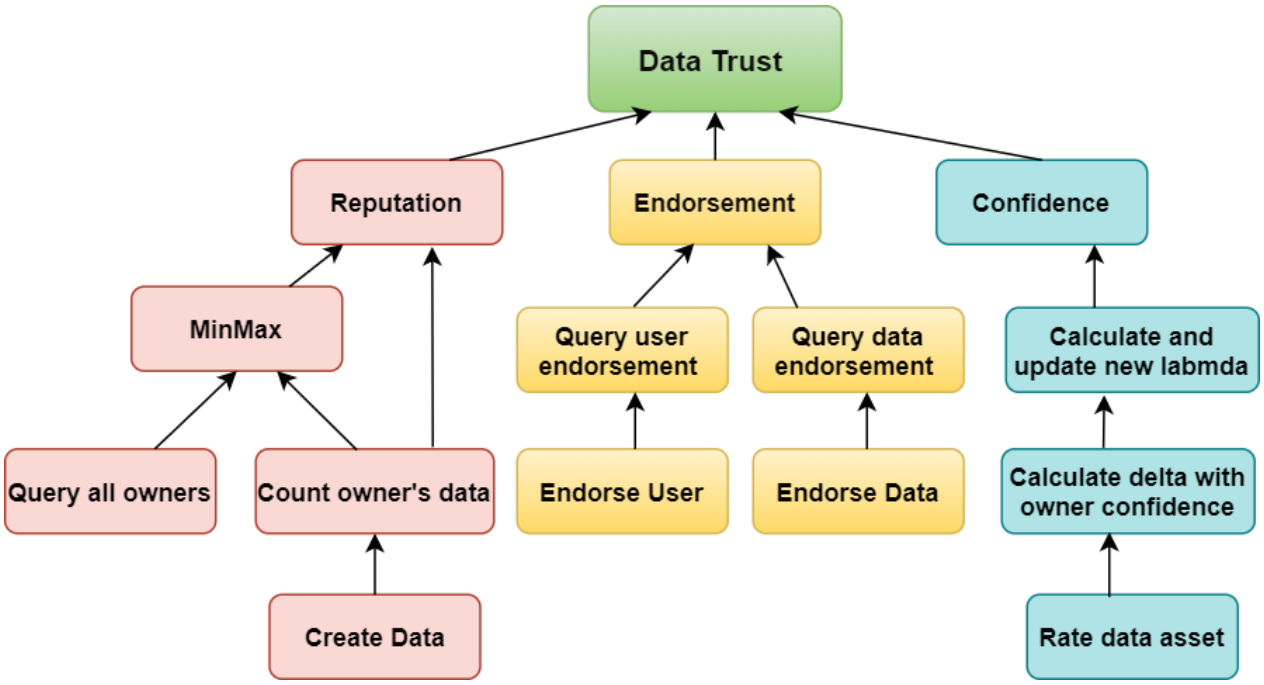

Figure 7.5: Smart contract transactions.

possible to generate customized and flexible queries.

7.5 System implementation

In this section, I discuss system implementation with regard to smart contracts design and implementation and adaptive endorsement. The program codes of the project implementation and evaluation experiments are available in the Github repository [1]. I have used Hyperledger Fabric version 2.2.0 for implementing the system. Hyperledger Fabric [13] is a well-known permissioned blockchain with a modular structure. This allows pluggability and customization. It also supports private communication and private data collections. Organizations can establish private communication by creating separate channels. Each organization can participate in separate channels so that it can develop multiple private communications. Each channel has a separate ledger, and the communications are restricted to the organizations within the channel. Moreover, the ledger can be privatized granularly to include only a particular set of participants.

7.5.1 Smart contract design

Figure 7.5 presents the transactions implemented in smart contract (chaincode) to assess data trust. As discussed in section refTrustModel, reputation, endorsement, and confidence are three main factors to asset the given data set's trustworthiness. Every data asset that I added to the system has multiple properties as

[1] https://github.com/sara-rouhani/dataTrust

following:

- dataID: a unique identifier for data assets.
- dataOwner: the userID that creates a data asset.
- hash: the hash value of a data asset stored in off-chain storage.
- owenrConfidence: the owners confidence in the provided data set. A float value between 0 to 1.
- endorsement: the number of endorsements that the data set is received. I have initialized it to 0.
- rating: the rating value that the data set is received from the users who studied the data set. A float number between 0 to 1.
- lambda: this is a factor discussed in formula 7.4, that indicates the effect of data owner's confidence. I have initialized it to 0.
- trust: the trust value that later will be calculated based on reputation, endorsement and confidence. It is initialized to null.

Reputation is based on the number of previous successful transactions that recorded a new data set to the system. This value will be calculated based on the ratio of the other users' reputations.

MinMax is a heavy computation transaction that queries all data assets stored on the ledger by dataOwners properties, counts the number of data for each owner's assets, and stores the minimum and the maximum number of owner's data stored on the ledger. Since MinMax is a heavy transaction, the MinMax transaction is not invoked to calculate the users' reputation. The system can regularly (for example, every hour) call the MinMax transaction and update the MinMax value on the ledger. Reputation transaction instead just queries the value of MinMax data stored on the ledger. This leads to significantly improving the performance of reputation transactions and, eventually, data trust transaction performance.

The endorsement is calculated based on two items, data asset endorsements and the owner of the data asset endorsements. Endorse User and Endorse data are two respective transactions that update data owners' endorsements and data assets' endorsements. Endorsement transaction calculates the data asset's endorsement based on querying the data related to the data asset endorsement and the owner of the data asset endorsement.

Every user who has studied the data asset can rate the quality of data by submitting a RateData transaction with a ratio between 0 to 1. Delta will be calculated based on the difference between the average ratings that the data asset has received and the owner's initial confidence. The new value for the lambda will be calculated and updated on the ledger accordingly. Confidence transaction queries the latest state for the lambda value for the data owner of the respective data asset. Eventually, data trust transaction invokes the three transactions, Reputation, Endorsement and Confidence, and returns each item's respective values.

7.5.2 Adaptive validation

Endorsement policies determine the smallest set of organizations with respect to their roles required to endorse and sign a transaction for it to be valid. The running peers refer to the endorsement policy to decide whether a transaction is valid. For example, OR('Organization1.member', AND('Organization2.peer', 'Organization3.peer')) is an endorsement policy that indicates either a member from organization1 must sign the transaction or one signature from a peer of the Organization2 MSP and one signature from a peer of the Organization3 MSP are required.

Hyperledger Fabric allows flexible endorsement at three different levels, chaincode level endorsement, collection-level endorsement and key-level endorsement [2]. Chaincode-level endorsements are agreed to by channel members. It means all transactions implemented in the chaincode follow the same endorsement policy. Collection-level endorsement policy sets the endorsement for a collection of data that are kept private in the channel. Key-level or stated-based endorsement provides a granular level, which we can utilize to achieve adaptive validation based on data owners' trustworthiness and data assets. In the key-level endorsement, we can implement a different endorsement policy for any single data stored on the ledger. This way, we can set a looser policy that requires fewer signatures for data assets that are higher trustworthy. We can set a tighter endorsement policy that requires more verifiers for less reliable data assets.

7.6 Evaluation

In this section, I present the result of the performance evaluation of the system. For implementing the system, I have used Hyperledger Fabric v2.2.0, and for measuring the system's performance, I have used Hyperledger Caliper v0.4.2. The system setup is based on 32GB memory and 4vCPU. The default network setup includes two organizations and two peers. For the last experiment, a new organization is added to the network. I initiated the ledger with 1000 data assets, randomly belonging to 10 participants, and their confidence value is a random number between 0.1 and 1. Alpha, beta, phi values have been assumed to be the same constant number for all participants. Table 7.2 describes the transactions that are invoked during test runs and their actions on the ledger.

Data owners create a new data asset on the ledger by invoking the CreateData transaction. 7.6 illustrates average latency, send rate and throughput for CreateData transaction in five rounds experiments, in which I have generated a different number of transactions in each round.

Figure 7.7 shows MinMax transaction average latency, send rate and throughput. MinMax is a heavy computational transaction that queries and counts all users' data assets and returns the minimum and maximum values. It stores these values on the ledger, which will be queried later for calculating data owners' reputations. This transaction can be regularly invoked to update the value of MinMaX. Therefore, Reputa-

[2] https://hyperledger-fabric.readthedocs.io/en/release-2.2

Table 7.2: Workload transactions.

Transaction	Description	Action
CreatData	Create a data asset belong to a random owner with random confidence	Write
MinMax	Queries all data assets stored on the ledger by dataOwners properties, counts the number of data for each owner's assets, and stores the minimum and the maximum number of owner's data stored on the ledger	Read-Write
Reputation	Calculates the reputation value for the input user by counting the ownser data and reading the MinMaX value form the ledger	Read
EndorUser	Read the stated of the data asset and updates the Endorsement value for the input user	Read-Write
EndorsData	Read the stated of the data asset and updates the Endorsement value for the input data	Read-Write
Endorsement	Calculate the endorsement value for the input data by querying the endorsement values belong to the data owner and data asset	Read
RateData	Read the stated of the data asset and add new ratring to the data asset	Read-Write
Lambda	Calculate the value of lambda based on delta (owner confidence and dataasset average rating), Update the new value for the lambda	Read- Write
DataTrust	Updates the value of Reputation, Endorsement, and Lambda (the effect on confidence) for the input data asset	Read-Write

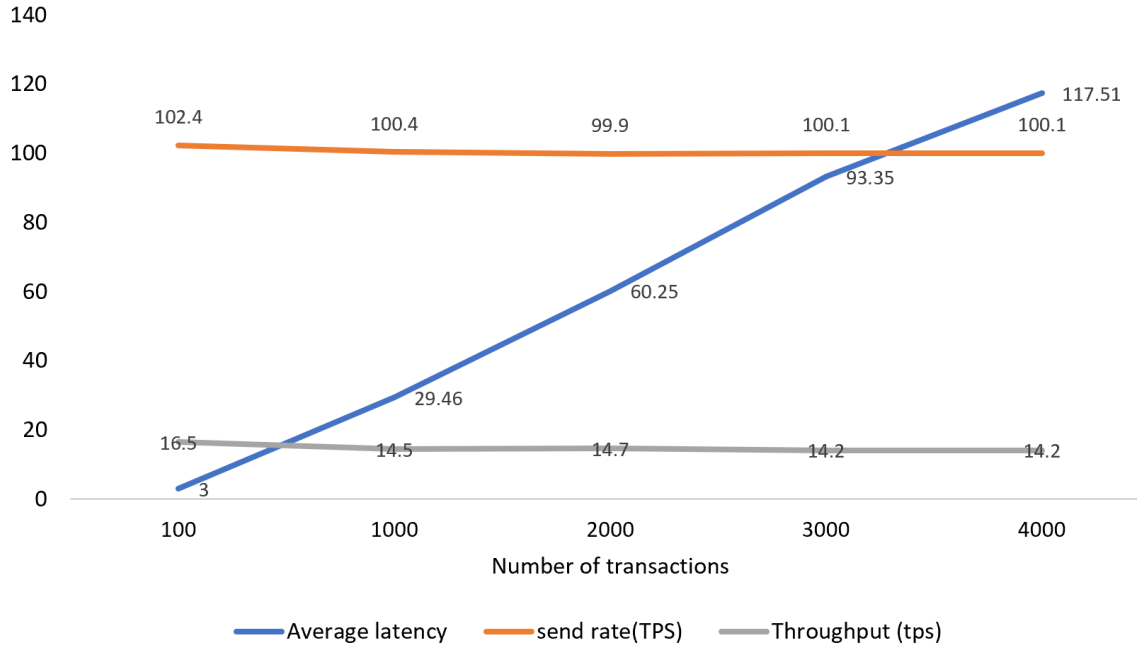

Figure 7.6: Send rate, throughput and average latency for CreateData transaction.

tion transactions and, consequently, DataTrust transactions do not overload with computing instantaneous MinMax values.

EndorsData and EndorsUser are two transactions that update the endorsement values for data assets and data owners. Ratedata also updates a data asset's rating by adding a new rating to the data asset rating array. Figures 7.8 and 7.9 shows the send rate, throughput and average latency for these transactions in five rounds experiments with a different number of transactions.

A comparison between send rate and throughput in all transactions involved in computing data trust are presented in figures 7.10 and 7.11 respectively. This evaluation is also based on five rounds with a different number of transactions. Confidence and Endorsement transactions present the highest send rate and throughput values as they query a single data asset and they calculate the Endorsement and Confidence

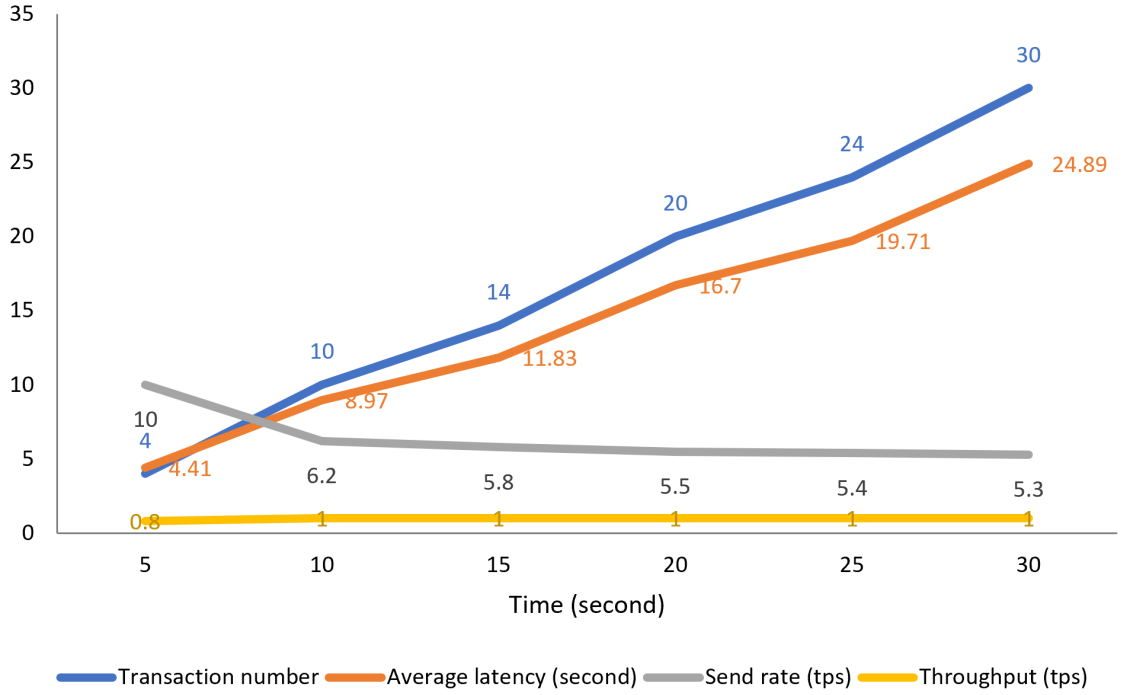

Figure 7.7: Calculate and record the minimum and maximum number of data assets belong to a single user (MinMax).

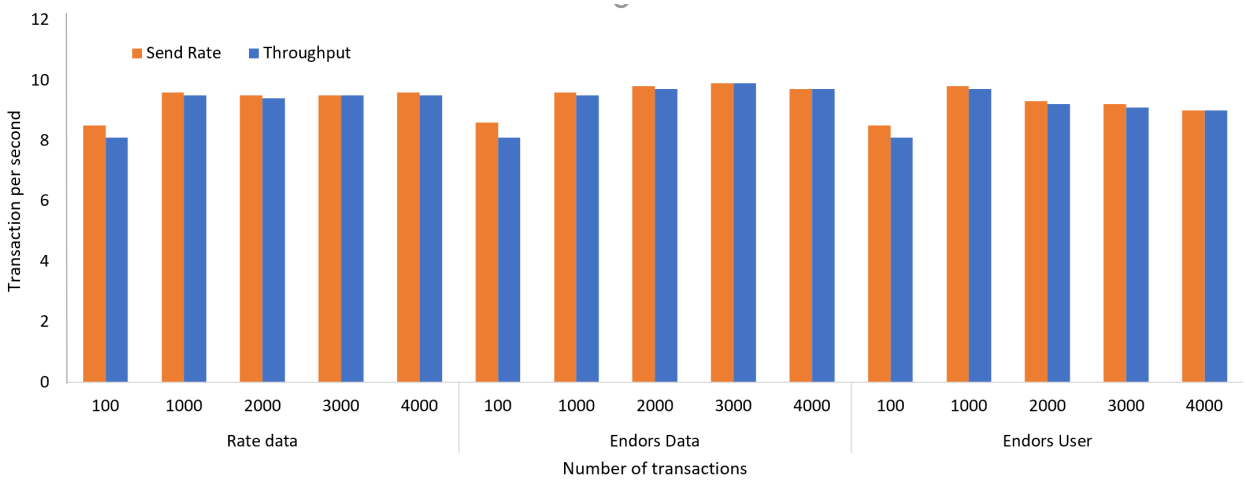

Figure 7.8: Sendrate and throughput for RateData, EndorsData, and EndorsUser transactions.

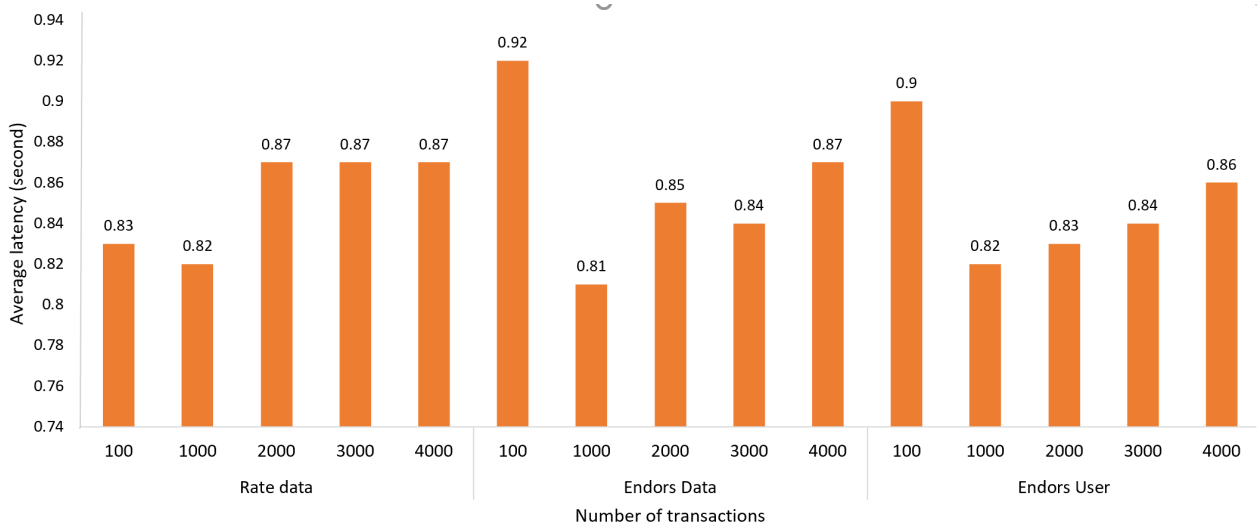

Figure 7.9: Average latency for RateData, EndorsData, and EndorsUser transactions.

based on data asset properties explained in section 7.5 (ownerConfidence, endorsement (both data and owner), rating, and lambda). CreateData transaction comparatively handles a high send rate (around 100 tps), but its throughput is between 16.4 to 14.2. EndorsData, EndorsUser, and RateData perform and update action on the ledger, and their send rate and throughput are around 10 tps. Reputation transaction has the lowest send rate and throughput, which are around 4.8 tps. Relatively, Reputation is a heavy transaction because it queries all data assets by filtering the data owner matches the current data owner.

Figure 7.13 shows the average latency for three main factors: reputation, endorsement, and confidence based on a different number of transactions. The confidence and endorsement are very fast transactions with a very low average latency because they only read a single value from the ledger. The reputation transaction is relatively a slower transaction. Although it does not have the overhead of MinMax transaction and only queries the latest value of the MinMax from the ledger, it still requires to query based on the number of data assets belonging to the current data owner. Therefore, it is a slower transaction comparing to the endorsement and confidence transactions.

Figure 7.14 presents the average latency for DataTrust transaction based on time. The left vertical axis indicates the number of successfully executed transactions during the given time, presented in the right vertical axis. The average latency for computing the data trust for 2502 transactions is 0.37 seconds that are completed in 500 seconds.

I have added a new organization and one new peer to the network. I have run the experiments on all transactions again. The comparison result between the average latency of all transactions in a network with two organizations and two peers and three organizations and three peers are presented in figure 7.15. As it is illustrated in the graph, adding a new organization does not affect the performance.

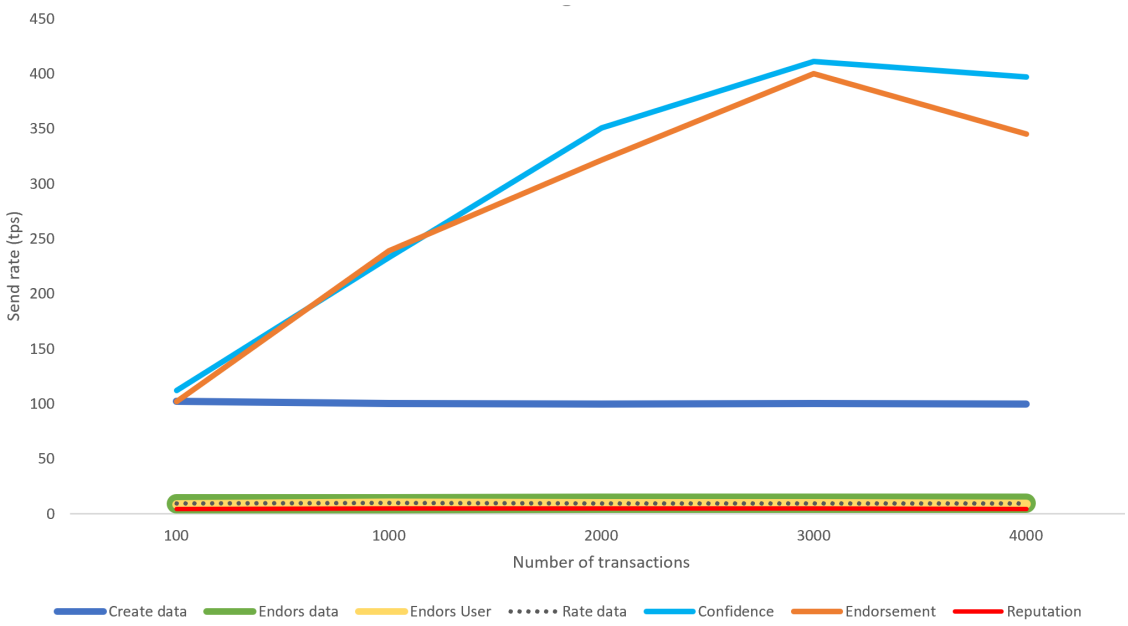

Figure 7.10: Send rate for all transaction.

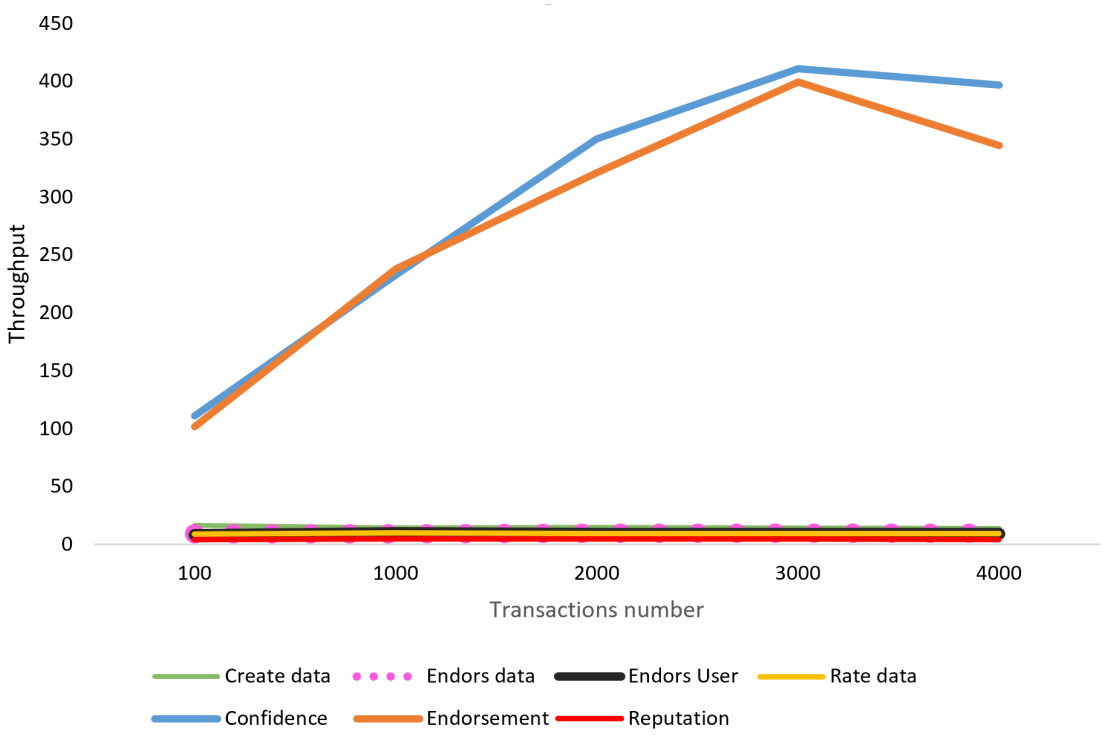

Figure 7.11: Throughput for all transaction.

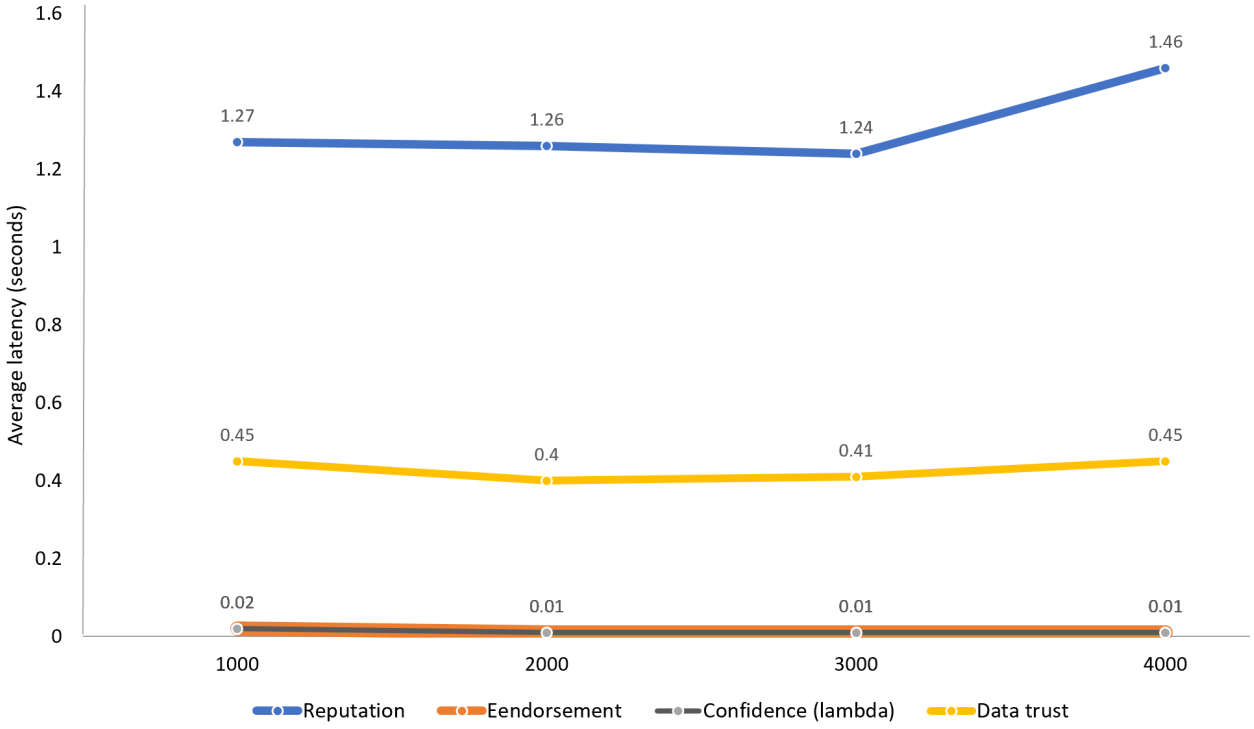

Figure 7.12: Average latency for DataTrust, Reputation, Endorsement and Confidence transactions.

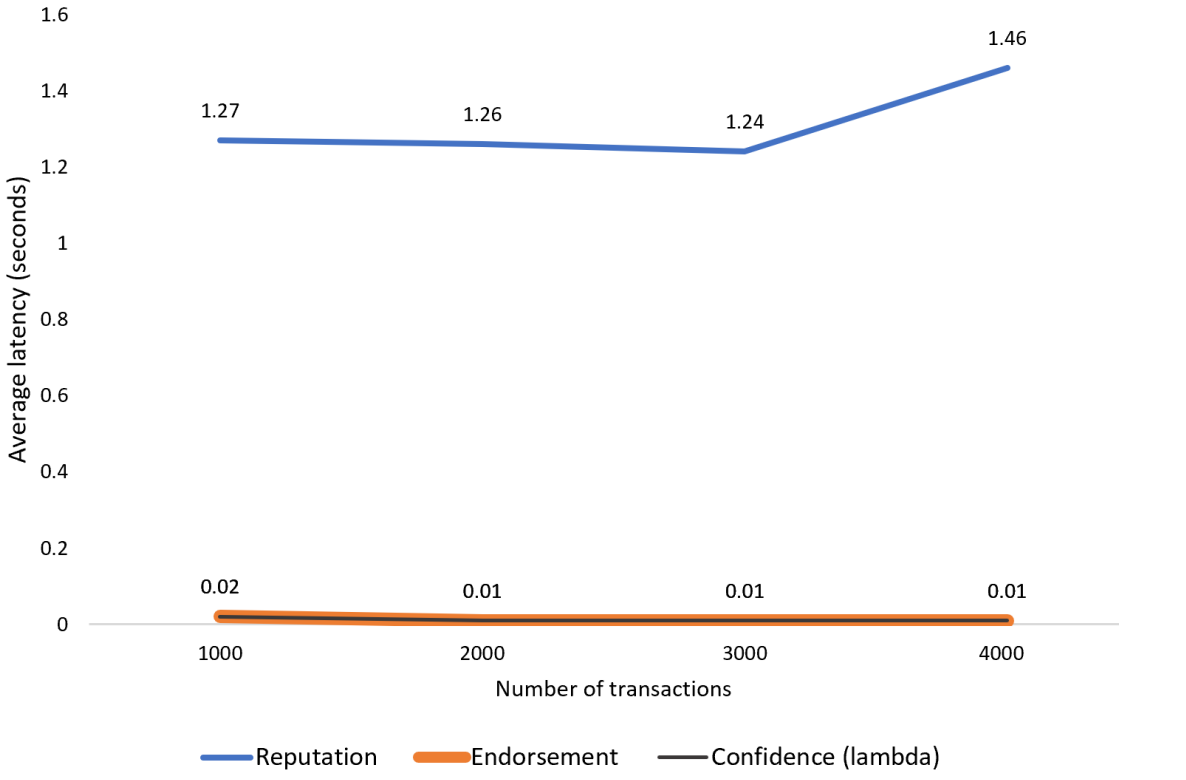

Figure 7.13: Average latency for three main factors: reputation, endorsement, and confidence based on different number of transactions.

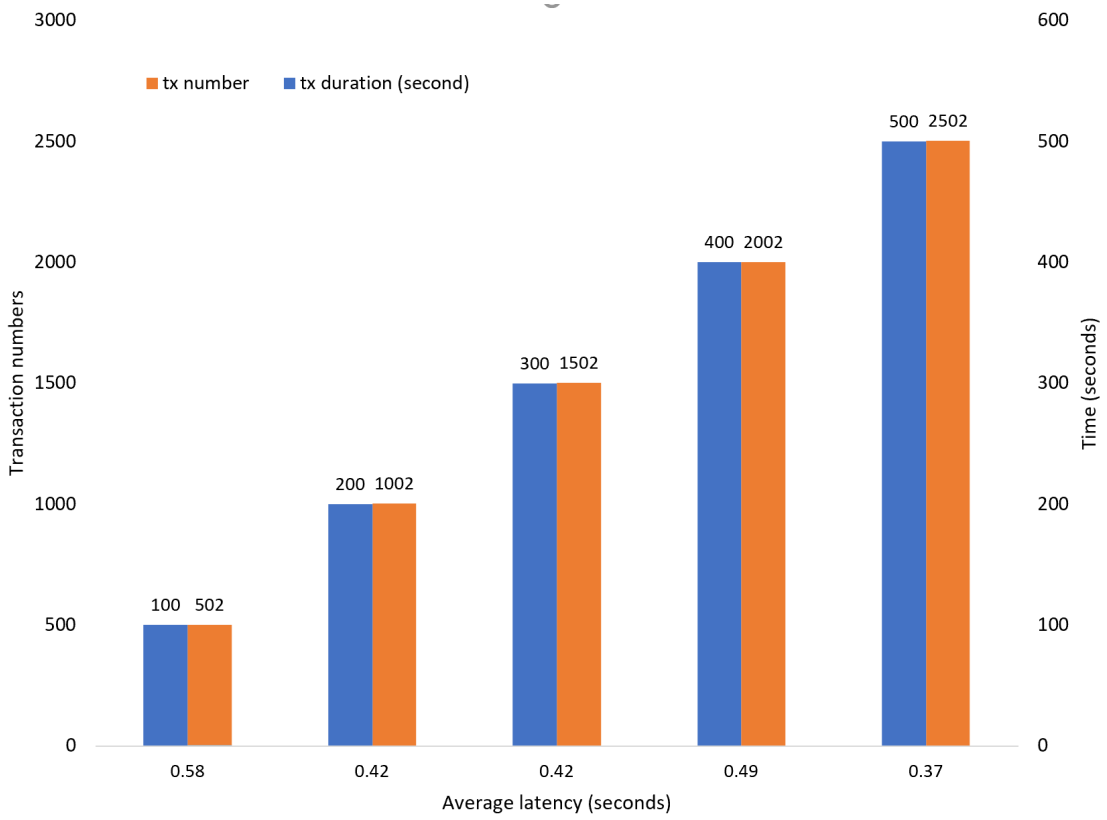

Figure 7.14: Average latency for DataTrust based on time.

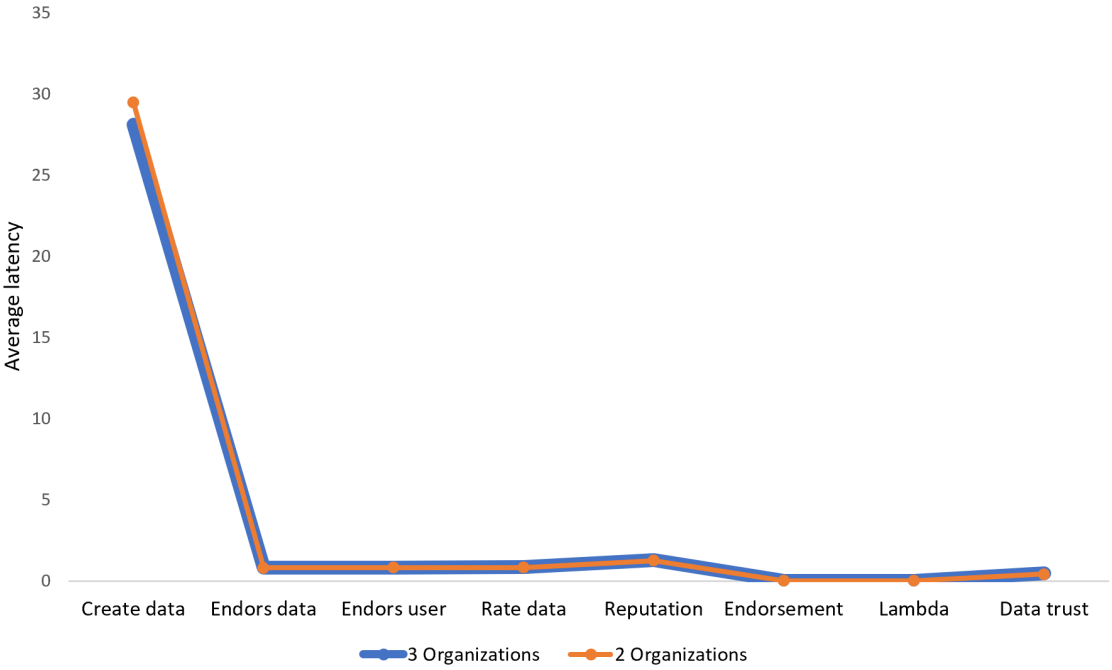

Figure 7.15: Increase the number of organizations.

7.7 Discussion

This section outlines how the proposed blockchain-based data trust framework enforces all the eight major requirements of data trust represented by O'Hara [124]. This scheme is shown to be sufficient, practical, and secure for trustworthy data sharing.

7.7.1 Discovery

I have used a permissioned blockchain for implementing the system. Unlike public blockchains, in permissioned blockchains, only authenticated stakeholders can interact with the blockchain and access the data recorded on the ledger. Desirability permissioned blockchains such as Hyperledger Fabric [13] provide a more granular level of access control for participants, so users joining blockchain can have various access restrictions to the different components of blockchain by associating policies. Authenticated data users are able to discover the available data assets, the properties of data sets represented as meta-data through the system interface.

The data's trustworthiness is also available for the data users through the trust value calculated by the proposed method. The details of each parameter involved in trust calculation for the represented data owner and data set are also available for potential data users. Besides, once the dataset is confirmed through a transaction validator, they will add a review to the data set and potential data users are allowed to access this review and discover the trustworthiness of the data set.

Most importantly, if a data user has previously studied the data set, the data user review regarding the trustworthiness of the data set is recorded on the ledger and available for future data users. It is an essential resource as the data users who studied and analyzed the data could bring the most accurate perspective on the trustworthiness of the data.

7.7.2 Provenance

Once the data owners add their data sets as data assets to the system, they must attach metadata related to the data provenance, such as the data origin, collection time, and collection method. This information can help both transaction verifiers and data users to assess the trustworthiness of the data.

Moreover, every time a data set is modified, an associated transaction is generated to update the data asset properties on the ledger. It helps to query data provenance and trace data evolution by identifying actual operations that have been performed on the data sets.

7.7.3 Access control

Data owners have full control over their data assets. They are the ones who decide on who gets access to their data, and by utilizing smart contracts, their access management is enforced automatically. Smart contracts

also enable fine-grained access checks to verify the authenticity of submitted transactions.

As I discussed in detail in section 7.4, data owners can set default users who can have access to their data sets or receive access requests from any data users. They inspect the request based on the purpose of the data users, and they decide to deny or accept the request and under which circumstances. The consent management smart contract records the consent between the data owner and data user on the ledger based on data owner specified conditions and possible penalties in case of data user violation.

7.7.4 Access

The data sets that include personal data must be de-identified or anonymized before sharing to ensure that individuals' interests are not compromised by providing access to their information. Besides, Hyperledger Fabric supports private data and private communication, which could be desirable for the data owners who do not want to expose the metadata associated with their data to all users of the system. They can use this feature and share their data assets info with their interested parties. We can also limit access to the data provenance can through customized policies in the smart contracts (chaincode). For example, data users can send requests to data owners to access reading the data set provenance.

7.7.5 Identity management

In Hyperledger Fabric as a permissioned blockchain, for every actor and user before starting interacting with the blockchain network, a digital identity is encapsulated in an X.509 digital certificate must be issued. This identity is essential to determine the correct permissions over resources and access to information recorded in a blockchain network. A digital identity can include additional attributes to specify the person or organization holding that identity. These attributes help data owners to identify those attempting to get access to their data assets.

7.7.6 Auditing

Auditing is one of the primary purposes of introducing blockchain as an infrastructure for implementing a data trust framework. Blockchain enables us to audit every process and interaction in the system. In the context of data sharing, blockchain provides an immutable audit trail of data modifications, access requests, access grants and revocations.

Data owners are able to query the history of transactions regarding access requests and modifications on access permissions on their data assets. Data users are also able to audit data assets origin and history of updates on the data sets. Furthermore, monitoring the immutable log of data transactions to automatically generate an audit trail and record any data breach attempts facilitates detecting possible threats [43].

7.7.7 Accountability

The proposed data trust framework with utilizing blockchain features increases transparency with respect to quality, access and usage of data. Once the data owners accept access requests to their data sets, data users become accountable for using the data under their control. The access enforcement point is an off-chain process that provides access to data sets based on the specified access limits represented by the data owner. The monitoring unit is responsible for tracking data users' actions. Monitoring the immutable log of actions generated by data users and detecting any violation or misuse can lead to penalty and recording permanently on the history of the data user on the ledger.

7.7.8 Impact

Identifying value, use, and misuse of data requires invoking data experts, who can participate as verifier nodes in the blockchain. The data owners can specify their data value and access conditions on their data assets, and they can be programmed into smart contracts. The penalty calculation must also be included in the consent management smart contract. The methods of measuring the impact of the data are out of the scope of this paper.

7.8 Chapter summary

In this chapter, I introduced an end-to-end data trust framework using permissioned blockchain and adaptive validation. The designed framework assesses the trustworthiness of input data using a novel trust model, including the data owner's reputation, endorsements and confidence in provided data. Therefore, the data users ensure that the available data set's trustworthiness has been adaptively examined and updated. Data owners also can benefit from secure, transparent, and automatic access management using smart contracts. They have full control over their data assets, and they are the only actors in the system who can regulate access permissions without relying on third parties. By employing blockchain's provenance and audibility, data owners can monitor the trace of access regulations and modifications on their data assets. Moreover, valuable logs can be queried from the ledger to provide a transparent view of the system, identify suspicious requests, and detect protocol breaches leading to discovering possible threads. Evaluation results indicate the system's effectiveness in handling a large number of transactions for writing, updating, and querying trust parameters value.

8 Conclusion and future direction

8.1 Research contributions

8.1.1 Literature reviews

Smart contracts are the main motive to utilize blockchain in various applications. I comprehensively reviewed the topic of smart contracts from different perspectives, including security, performance, and application design [140].

- I systematically examined the key concepts and proposed recent studies and developments, and I presented the method of collecting studies. I selected the related studies and provided a systematic mapping method to present an extensive review of smart contracts in distributed ledger technology.

- I discussed the supporting platforms for smart contracts and compared them. It introduces smart contract challenges and limitations, and the presented solutions to address those challenges are discussed and compared.

- I discussed the existing security problems, and various potential solutions, including formal verification, code analyzer tools, secure consensus methods, privacy-enhancing, and secure programming language.

- I investigated the performance analyzing frameworks, real-time monitoring tools and potential solutions such as concurrent and execution of transactions.

- I investigated the state of the decentralized applications with a focus on smart contracts. I also discussed the research gap and future direction.

Many blockchain-based decentralized applications have utilized blockchain for decentralized access control to resolve centralized access control systems' problems. I reviewed the blockchain-based applications focusing on access control [144]. I declared state of the art and the challenges of blockchain-based access control systems in this study.

First, I introduced the problems of centralized access control systems. I studied the potential and limitations of blockchain technology to address these problems. Second, I presented an overview of access control studies and blockchain-based access control platforms.

8.1.2 Investigated and developed distributed data sharing and access control in particular fields

- Medical data fields:

  In current medical systems, patients' data stores scattered in various unlinked systems and access to the patient data could be time-consuming and inefficient. On the other hand, patients do not have any control over who has access to their data and its purpose. I designed and implemented a patient-centric system for sharing medical data using permissioned blockchain [142] as a Mitacs project.

  In this application, I proposed a method to store stored medical data in a secure cloud off-chain repository and the hash of the data and access rules on a permission blockchain. By employing smart contracts, patients are able to grant or revoke access permissions directly by submitting the respective transaction. The presented system utilizes blockcahin's audibility features to provide patients with access traces, including the history of users who have been granted or revoked access to their medical data and their permissions on patient's data over time. Connecting the off-chain and on-chain application is the novel part of this study and one of the first practical implementations for such an application scenario.

- Business process management field:

  Lack of trust and divided data are the two main drawbacks of centrally controlled business processes. I have collaborated in investigating the applicability of executing a real-world untrusted business process on the permissioned blockchain to address the integration and trust issues. [135]. This study has remarked on the feasibility of the execution of Order Processing as a case of a real-world business process on a permissioned blockchain.

- Physical access control field:

  Most access control and authorization studies have focused on logical access control because of the greater attention to the data's importance in recent years. However, the physical access control system's functionality is not limited to the operation of hardware devices; instead, the proper operation of the software layer and its process are significant as well.

  I proposed and implemented a novel distributed application for the software layer of physical access control systems [145] to prevent possible threads arising from centralized administrative systems. I implemented a proof of concept of such an application using Hyperledger Composer, completed with a simulated scenario to test the system indicating the proposed approach's feasibility. Performance analysis discussions on results generated by Hyperledger Caliper is also presented.

8.1.3 Designed, developed, and analysed distributed access control

After presenting blockchain-based systems to address data sharing and access control in particular domains, such as healthcare, business process, and physical access control, I proposed a distributed access control system that does not bind to a particular application domain. The suggested solution is applicable for any application with multi-stakeholder with different incentives.

My research presents a distributed ABAC system based on blockchain to provide trusted auditing of access attempts. Blockchain records a tamper-proof history of all changes to the access policies, attributes, and access attempts and their results. This trace could be utilized as non-repudiation and useful logging to detect potential systems threats. Besides auditability, my research offers a level of transparency that both access requesters and resource owners can benefit from.

I validated the proposed solution through a use case of independent digital libraries. I provided a detailed performance analysis of the implementation. The evaluation result compares the proposed solution's efficiency under different consensus mechanisms, Raft and Kafka, different databases, CouchDB and GoLelelD, and customizable network configuration. This work is conducted as a part of my internship program with the Linux Foundation (Hyperledger) and their blockchain project.

8.1.4 Designed, developed, and analysed distributed end-to-end data trust

Utilizing my previous studies' foundation, I presented an end-to-end framework for data trust using blockchain technology to facilitate data sharing and trustworthiness control.

- My research presents a trust model to assess the trustworthiness of data provided by data owners that apply adaptive validation and employ smart contracts to regulate the accesses permissions and consent management for sharing data with data users securely.
- This framework addresses both data owners and data users' concerns by ensuring the trustworthiness and quality of the data at origin and ethical and secure usage of the data after sharing the data.
- The presented framework adaptively assesses the trustworthiness of input data using a novel trust model, including the data owner's reputation, endorsements and confidence in provided data.
- Data owners also can help with secure, transparent, and automatic access management implemented by smart contracts. They have complete control over their data assets by presenting them as the only actors in the system who can manage access permissions without relying on third parties.

8.2 Future work

For the future direction, I am looking toward expanding the blockchain infrastructure for implementing data trust applications by providing interoperably between various blockchain platforms. There has been a

significant advance in implementing various and solitude blockchain platforms to utilize blockchain capabilities in multiple areas. However, this has led to creating numerous incompatible blockchain platforms with the inability to connect and work interoperably with other blockchain platforms. For example, Hyperledger Indy [1] provides a robust infrastructure for digital identities based on Blockchain. At the same time, Hyperledger Fabric is a permissioned blockchain that provides a modular and adaptable design fitting a broad range of industry use cases. By establishing a secure and reliable connection between these two blockchain platforms, we can leverage the capability of both of them.

Interoperability is a crucial requirement for the survivability and manageability of blockchain systems. For blockchain systems to become predominant, system architectures should not concern about choosing the right platform, smart contracts programming languages. Similarly, the back end of the web does not matter to the web application designers. To meet this objective, the interoperability between multiple blockchain platforms should be provided. There are three directions in studying the possibility of blockchain interoperability, cryptocurrency-directed, blockchain engines and blockchain connectors [18, 26].

Cryptocurrency-directed is a category that addresses the interoperability between public blockchains that support cryptocurrencies. Blockchain engines are frameworks that create reusable data, network, consensus, incentive, and smart contract layers to create customized blockchains for general use-cases and heterogeneous systems [18]. There are two blockchain interoperability initiatives Cosmos [2] and Polkadot [3] that implement blockchain interoperability through blockchain engines. The Blockchain Connector includes interoperability solutions that belong to the neither of cryptocurrency-directed nor blockchain engines. Several other blockchain interoperability solutions such as trusted relays, blockchain agnostic protocols, blockchain of blockchains, and blockchain migrators are studied in this category.

We have already implemented [4] a successful prototype of blockchain interoperability based on Publish/-Subscribe architecture. Publish/Subscribe, or shortly pub/sub, is a distributed interaction pattern that describes the flow of messages between publishers and subscribers. So, neither the publisher needs to know anything about recipients of their data (subscribers), nor subscribers need to know the number and location of publishers. This is referred to as loose coupling feature in pub/sub systems. Since there is no direct interaction between publishers and subscribers, and brokers are the entities that control the interactions between publishers and subscribers, as shown in figure 8.1. The publish/subscribe messaging pattern is well adopted in many application areas such as IoT, mobile environments, military systems, and peer to peer networks to improve performance, reliability and scalability.

We proposed a solution in which a blockchain acts as a messaging broker and keeps track of all the topics provided by other blockchain networks. Publisher blockchains can send messages to the topics, and the subscriber blockchains will be automatically notified immediately by any changes in the topic through a received

[1] https://www.hyperledger.org/use/hyperledger-indy
[2] https://cosmos.network/
[3] https://polkadot.network/
[4] github.com/hyperledger-labs/pubsub-interop

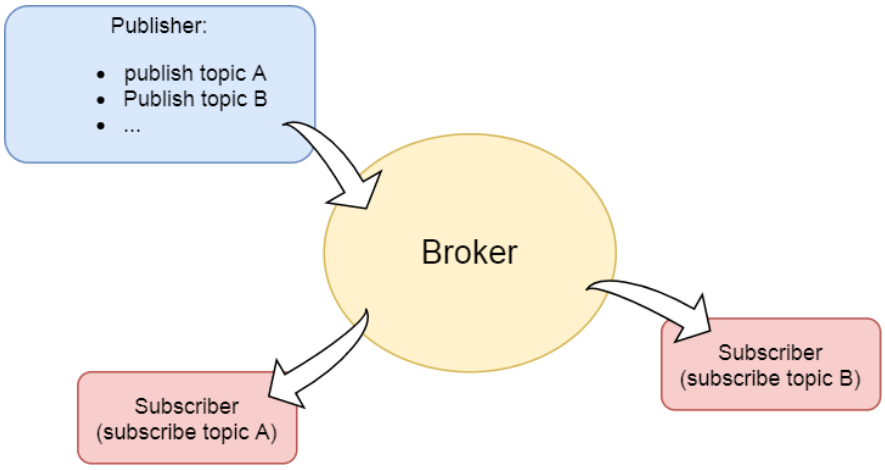

Figure 8.1: Publish-Subscribe architecture.

publish request. We look for connecting heterogeneous blockchain platforms with different architecture and infrastructure through the presented solution.

The proposed platform has three main components: broker blockchain, subscriber blockchains, and publisher blockchain as it is shown in fig 8.2. Every component runs by a different blockchain platform to support the interoperability.

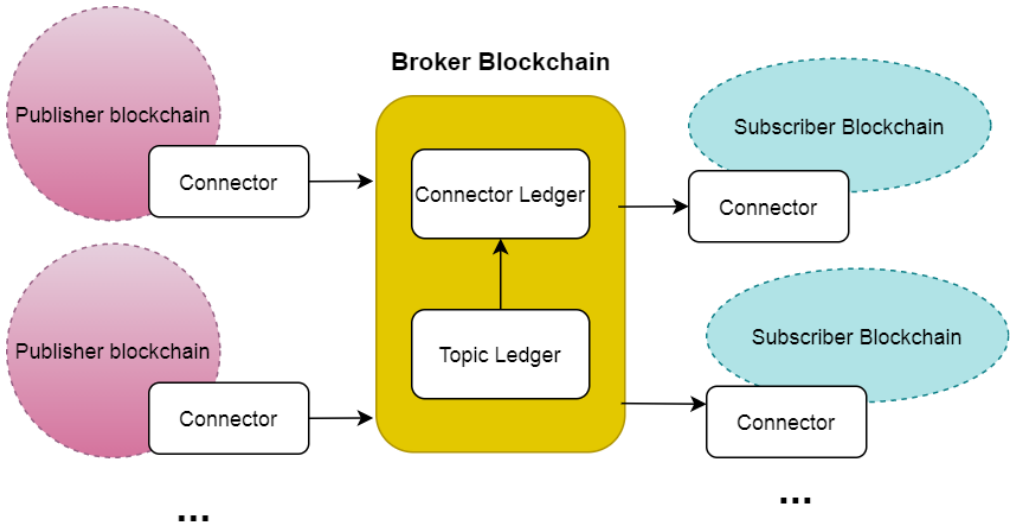

Figure 8.2: Publish-Subscribe architecture for blockchain interoperability.

Although pub/sub is a decoupled model, providing a secure infrastructure to prevent illegal information flow among connected blockchains is required as each blockchain can run by a network of nodes with different trust levels.

Hyperledger Fabric version 2 is used for the broker and publisher and Hyperledger Besu and Hyperledger Fabric version 1.4 as examples of subscribers for implementing the system. Figure 8.3 shows The system throughput and average latency for various functionalities throughout time by changing the send rate. The

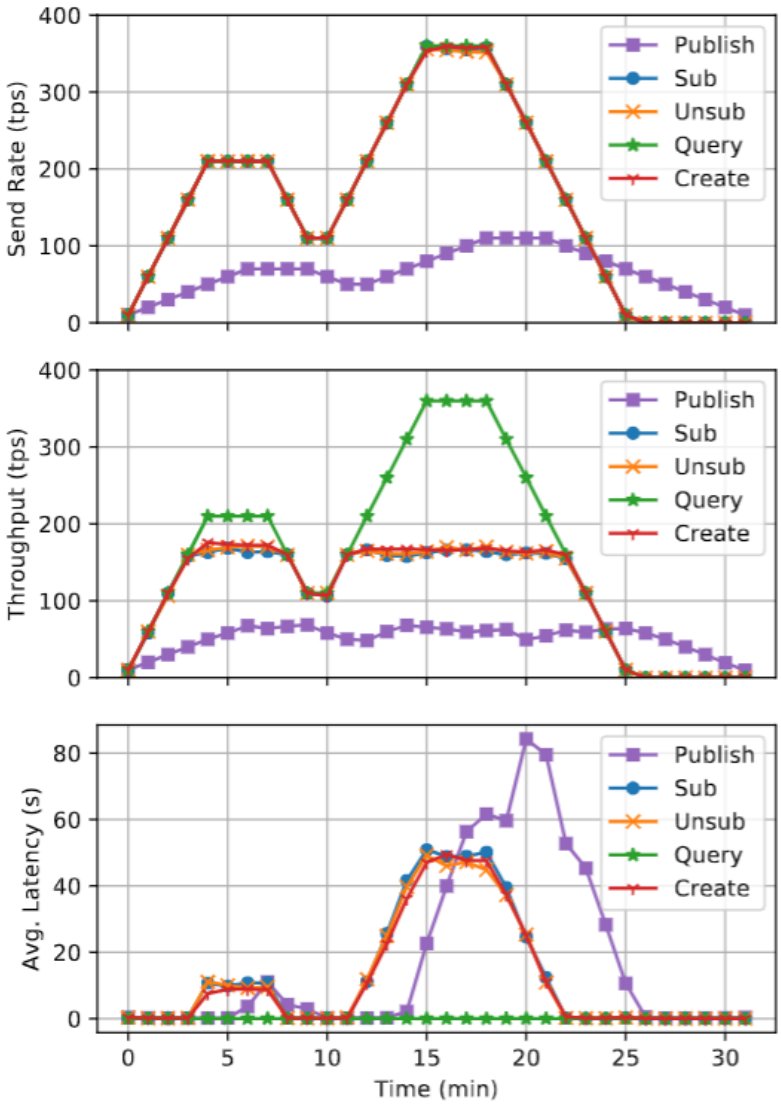

Figure 8.3: The trend of system throughput and average latency for various functionalities throughout time with the change of request send rate [63].

words publish, sub, unsub, query, and create in the plots stand for PublishToTopic, SubscribeToTopic, UnsubscribeFromTopic, QueryTopic, and CreateTopic functions, respectively.

In the presented platform prototype, every participant can subscribe to all existing topics, and there is no access control mechanism implemented. For the future work, the private data feature offered by Hyperledger Fabric can be employed to establish private channels in the broker blockchain and keep data topics separate from other organizations. Furthermore, an access control mechanism can enhance the pub/sub system's security and privacy to control data flow at a more granular level.

As the next step, we can regulate the current architecture for implementing a data trust application connecting the blockchain platforms that offer necessities fitting the application, such as digital identity for user authentication and modular structure for adaptive, efficient consensus mechanism and databases.

8.3 Conclusion

To conclude, this dissertation contributes to the area of decentralized access control and data trust using blockchain technology and smart contracts to achieve transparency, data provenance, reliability and automatic operations.

Data trust in the context of three domains including sharing healthcare data , executing and monitoring business processes and physical access control has been investigated and implemented.

I have designed and developed a patient-centric system to manage medical data without involving third parties efficiently. Patients have the control to share their data with any interested party. They can transparently trace the access control transactions and see the list of users who currently have access to their data and the users who had access in the past.

A case study of a real world business process has been implemented on a permissioned blockchain. Several untrusted parties can collaborate through the business rules, sequence of logics and automatic process checking embedded in smart contracts, while their sensitive information remains private from their competitors. The result confirms that the deployed business network on the blockchain ultimately ensures the approved transactions' correctness and provides guaranteed access control over the network's business data.

An access control system has been developed for the software layer of physical access control systems using permissioned blockchain. I have implemented two access control models to provide both static and dynamic access regulations using a role-based access control model implemented by ACL and a rule-based access control model implemented by smart contracts. The system provides a reliable and tamper-proof transactions' log to query, and it could be accessible through authenticated and authorized users.

A decentralized ABAC access control model has been implemented using Hyperledger Fabric permissioned blockchain. A policy based architecture compromising three components for managing attributes and policies on the ledger and calculating access decisions have been implemented as smart contracts. A comprehensive performance analysis have been conducted utilizing the modular structure of Hyperledger Fabric, comparing the system performance based on different consensus mechanisms, different databases and various network configurations. The experimental evaluation results shows that the presented system can effectively handle a transaction throughput of 270 transactions per second, with an average latency of 0.54 seconds per transaction.

Finally, I designed and implemented an end to end data trust framework using blockchain technology and adaptive transaction validation. A trust model has been presented that assesses the provided data set's trustworthiness based on reputation, endorsement and data owner confidence. Therefore, the data users can assess the trustworthiness of available data sets. Data owners can serve with a secure, transparent, and automatic access management implemented by smart contracts. They have complete control over their data assets, and they are the only actors in the system who can regulate access permissions without relying on third parties. Moreover, important logs can be queried from the ledger to present a transparent view of the

system, identify suspicious requests, and detect protocol breaches leading to identifying potential threats. I have conducted a comprehensive experiment to examine the system performance and scalability. Evaluation results show the system's effectiveness in handling a large number of transactions for writing, updating, and querying trust parameters value.

This dissertation's main contribution includes developing and evaluating decentralized access control systems based on multiple application domains and various access control methods as a critical component of data stewardship. Consequently, developing a reliable data trust framework that addresses the trust issue in both data owners and data users.

References

[1] Survey: Patients see 18.7 different doctors on average, 2010. https://www.prnewswire.com/news-releases/survey-patients-see-187-different-doctors-on-average-92171874.html, (accessed Feb 9, 2020).

[2] Istanbul byzantine fault tolerant consensus protocol, 2017. https://github.com/ethereum/EIPs/issues/650, (accessed March 01, 2019).

[3] Nem technical reference, 2018. https://nem.io/wp-content/themes/nem/files/NEM_techRef.pdf, (accessed March 01, 2019).

[4] Nabil R Adam, Vijayalakshmi Atluri, Elisa Bertino, and Elena Ferrari. A content-based authorization model for digital libraries. *IEEE Transactions on knowledge and data engineering*, 14(2):296–315, 2002.

[5] Mustafa Al-Bassam. Scpki: A smart contract-based pki and identity system. In *Proceedings of the ACM Workshop on Blockchain, Cryptocurrencies and Contracts*, pages 35–40, 2017.

[6] Shorouq Alansari, Federica Paci, and Vladimiro Sassone. A Distributed Access Control System for Cloud Federations. *Proceedings - International Conference on Distributed Computing Systems*, pages 2131–2136, 2017.

[7] Elvira Albert, Pablo Gordillo, Benjamin Livshits, Albert Rubio, and Ilya Sergey. EthIR: A Framework for High-Level Analysis of Ethereum Bytecode. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11138 LNCS:513–520, 2018.

[8] Elvira Albert, Pablo Gordillo, Albert Rubio, and Ilya Sergey. Running on fumes-preventing outof-gas vulnerabilities in ethereum smart contracts using static resource analysis. *VECoS, LNCS. Springer*, 2019.

[9] Arwah Alsaad, Kieron O'Hara, and Leslie Carr. Institutional repositories as a data trust infrastructure. In *Companion Publication of the 10th ACM Conference on Web Science*, pages 1–4, 2019.

[10] Sidney Amani, Myriam Bégel, Maksym Bortin, and Mark Staples. Towards verifying ethereum smart contract bytecode in Isabelle/HOL. *Proceedings of the 7th ACM SIGPLAN International Conference on Certified Programs and Proofs - CPP 2018*, 61(ii):66–77, 2018.

[11] Jian An, Jindong Cheng, Xiaolin Gui, Wendong Zhang, Danwei Liang, Ruowei Gui, Lin Jiang, and Dong Liao. A lightweight blockchain-based model for data quality assessment in crowdsensing. *IEEE Transactions on Computational Social Systems*, 7(1):84–97, 2020.

[12] Anne Anderson, Bill Parducci, and Carlisle Adams. OASIS eXtensible Access Control Markup Language (XACML), 2006. http://research.sun.com/projects/xacml/XMLCOP{_}060620{_}slides.pdf, (accessed Feb 9, 2020).

[13] Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich, et al. Hyperledger fabric: a distributed operating system for permissioned blockchains. In *Proceedings of the Thirteenth EuroSys Conference*, pages 1–15, 2018.

[14] Parwat Singh Anjana, Sweta Kumari, Sathya Peri, Sachin Rathor, and Archit Somani. An Efficient Framework for Optimistic Concurrent Execution of Smart Contracts. *Proceedings - 27th Euromicro International Conference on Parallel, Distributed and Network-Based Processing, PDP 2019*, pages 83–92, 2019.

[15] Asaph Azaria, Ariel Ekblaw, Thiago Vieira, and Andrew Lippman. MedRec: Using blockchain for medical data access and permission management. *Proceedings - 2016 2nd International Conference on Open and Big Data, OBD 2016*, pages 25–30, 2016.

[16] Massimo Bartoletti and Roberto Zunino. Bitml: a calculus for bitcoin smart contracts. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 83–100. ACM, 2018.

[17] Rafael Belchior, André Vasconcelos, and Miguel Correia. Towards Secure, Decentralized, and Automatic Audits with Blockchain. In *European Conference on Information Systems*, 2020.

[18] Rafael Belchior, André Vasconcelos, Sérgio Guerreiro, and Miguel Correia. A survey on blockchain interoperability: Past, present, and future trends. *arXiv preprint arXiv:2005.14282*, 2020.

[19] Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, Eran Tromer, and Madars Virza. Snarks for c: Verifying program executions succinctly and in zero knowledge. In *Advances in Cryptology–CRYPTO 2013*, pages 90–108. Springer, 2013.

[20] K. Bhargavan, A. Delignat-Lavaud, C. Fournet, A. Gollamudi, G. Gonthier, N. Kobeissi, N. Kulatova, A. Rastogi, T. Sibut-Pinote, N. Swamy, and S. Zanella-Béguelin. Formal verification of smart contracts: Short paper. *PLAS 2016 - Proceedings of the 2016 ACM Workshop on Programming Languages and Analysis for Security, co-located with CCS 2016*, pages 91–96, 2016.

[21] Thomas Bocek, Bruno B Rodrigues, Tim Strasser, and Burkhard Stiller. Blockchains everywhere-a use-case of blockchains in the pharma supply-chain. In *Integrated Network and Service Management (IM), 2017 IFIP/IEEE Symposium on*, pages 772–777. IEEE, 2017.

[22] Gilad Bracha and David M. Ungar. Mirrors: design principles for meta-level facilities of object-oriented programming languages. In *OOPSLA*, 2004.

[23] Santiago Bragagnolo, Henrique Rocha, Marcus Denker, and Stéphane Ducasse. Smartinspect: solidity smart contract inspector. In *2018 International Workshop on Blockchain Oriented Software Engineering (IWBOSE)*, pages 9–18. IEEE, 2018.

[24] António Brandão, Henrique São Mamede, and Ramiro Gonçalves. Systematic review of the literature, research on blockchain technology as support to the trust model proposed applied to smart places. In *World Conference on Information Systems and Technologies*, pages 1163–1174. Springer, 2018.

[25] Lexi Brent, Anton Jurisevic, Michael Kong, Eric Liu, Francois Gauthier, Vincent Gramoli, Ralph Holz, and Bernhard Scholz. Vandal: A scalable security analysis framework for smart contracts. *arXiv preprint arXiv:1809.03981*, 2018.

[26] Vitalik Buterin. Chain interoperability. *R3 Research Paper*, 2016.

[27] Vitalik Buterin. Critical update re: Dao vulnerability. *Ethereum Blog, June*, 2016. https://blog.ethereum.org/2016/06/17/critical-update-re-dao-vulnerability/, (accessed July 17, 2018).

[28] Vitalik Buterin et al. A next-generation smart contract and decentralized application platform. *white paper*, 2014.

[29] Vitalik Buterin, G Wood, V Zamfir, and J Coleman. Notes on scalable blockchain protocols. *Ethereum Foundation*, 31, 2015.

[30] Michael J Cahill, Uwe Röhm, and Alan D Fekete. Serializable isolation for snapshot databases. *ACM Transactions on Database Systems (TODS)*, 34(4):20, 2009.

[31] Seraphin Calo, Dinesh Verma, Supriyo Chakraborty, Elisa Bertino, Emil Lupu, and Gregory Cirincione. Self-generation of access control policies. In *Proceedings of the 23nd ACM on Symposium on Access Control Models and Technologies*, pages 39–47, 2018.

[32] Cinzia Cappiello, Marco Comuzzi, Florian Daniel, and Giovanni Meroni. Data quality control in blockchain applications. In *International Conference on Business Process Management*, pages 166–181. Springer, 2019.

[33] Roberto Casado-Vara, Fernando de la Prieta, Javier Prieto, and Juan M Corchado. Blockchain framework for iot data quality via edge computing. In *Proceedings of the 1st Workshop on Blockchain-enabled Networked Sensor Systems*, pages 19–24, 2018.

[34] Silvana Castano, Mariagrazia Fugini, Giancarlo Martella, and Pierangela Samarati. *Database Securi*. Addison—Wesley, 1995.

[35] Miguel Castro and Barbara Liskov. Practical byzantine fault tolerance and proactive recovery. *ACM Transactions on Computer Systems (TOCS)*, 20(4):398–461, 2002.

[36] Shi-Cho Cha, Jyun-Fu Chen, Chunhua Su, and Kuo-Hui Yeh. A blockchain connected gateway for ble-based devices in the internet of things. *IEEE Access*, 6:24639–24649, 2018.

[37] Si Chen, Rui Shi, Zhuangyu Ren, Jiaqi Yan, Yani Shi, and Jinyu Zhang. A blockchain-based supply chain quality management framework. In *e-Business Engineering (ICEBE), 2017 IEEE 14th International Conference on*, pages 172–176. IEEE, 2017.

[38] Ting Chen, Xiaoqi Li, Xiapu Luo, and Xiaosong Zhang. Under-optimized smart contracts devour your money. In *2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pages 442–446. IEEE, 2017.

[39] Wuhui Chen, Yufei Chen, Xu Chen, and Zibin Zheng. Toward secure data sharing for the iov: a quality-driven incentive mechanism with on-chain and off-chain guarantees. *IEEE Internet of Things Journal*, 7(3):1625–1640, 2019.

[40] Yi-Hui Chen, Shih-Hsin Chen, and Iuon-Chang Lin. Blockchain based smart contract for bidding system. In *2018 IEEE International Conference on Applied System Invention (ICASI)*, pages 208–211. IEEE, 2018.

[41] Raymond Cheng, Fan Zhang, Jernej Kos, Warren He, Nicholas Hynes, Noah Johnson, Ari Juels, Andrew Miller, and Dawn Song. Ekiden: A platform for confidentiality-preserving, trustworthy, and performant smart contract execution. *arXiv preprint arXiv:1804.05141*, 2018.

[42] Jin-Hee Cho, Kevin Chan, and Sibel Adali. A survey on trust modeling. *ACM Computing Surveys (CSUR)*, 48(2):1–40, 2015.

[43] Olivia Choudhury, Issa Sylla, Noor Fairoza, and Amar Das. A blockchain framework for ensuring data quality in multi-organizational clinical trials. In *2019 IEEE International Conference on Healthcare Informatics (ICHI)*, pages 1–9. IEEE, 2019.

[44] Allen Clement, Harry Li, Jeff Napper, Jean-Philippe Martin, Lorenzo Alvisi, and Mike Dahlin. Bar primer. In *2008 IEEE International Conference on Dependable Systems and Networks With FTCS and DCC (DSN)*, pages 287–296. IEEE, 2008.

[45] Brian F Cooper, Adam Silberstein, Erwin Tam, Raghu Ramakrishnan, and Russell Sears. Benchmarking cloud serving systems with ycsb. In *Proceedings of the 1st ACM symposium on Cloud computing*, pages 143–154. ACM, 2010.

[46] Jason Paul Cruz, Yuichi Kaji, and Naoto Yanai. RBAC-SC: Role-based access control using smart contract. *IEEE Access*, 6:12240–12251, 2018.

[47] Gaby G. Dagher, Jordan Mohler, Matea Milojkovic, and Praneeth Babu Marella. Ancile: Privacy-preserving framework for access control and interoperability of electronic health records using blockchain technology. *Sustainable Cities and Society*, 39(December 2017):283–297, 2018.

[48] Gabriele D'Angelo, Stefano Ferretti, and Moreno Marzolla. A blockchain-based flight data recorder for cloud accountability. In *Proceedings of the 1st Workshop on Cryptocurrencies and Blockchains for Distributed Systems*, pages 93–98. ACM, 2018.

[49] Chris Dannen. *Introducing Ethereum and Solidity*. Springer, 2017.

[50] Casimer DeCusatis, Marcus Zimmermann, and Anthony Sager. Identity-based network security for commercial blockchain services. In *Computing and Communication Workshop and Conference (CCWC), 2018 IEEE 8th Annual*, pages 474–477. IEEE, 2018.

[51] Volkan Dedeoglu, Raja Jurdak, Guntur D Putra, Ali Dorri, and Salil S Kanhere. A trust architecture for blockchain in iot. In *Proceedings of the 16th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, pages 190–199, 2019.

[52] Kevin Delmolino, Mitchell Arnett, Ahmed E Kosba, Andrew Miller, and Elaine Shi. Step by Step Towards Creating a Safe Smart Contract: Lessons and Insights from a Cryptocurrency Lab. *IACR Cryptology ePrint Archive*, page 460, 2015.

[53] Thomas Dickerson, Paul Gazzillo, Maurice Herlihy, and Eric Koskinen. Adding concurrency to smart contracts. *Distributed Computing*, pages 1–17, 2019.

[54] Sheng Ding, Jin Cao, Chen Li, Kai Fan, and Hui Li. A Novel Attribute-Based Access Control Scheme Using Blockchain for IoT. *IEEE Access*, 7:38431–38441, 2019.

[55] Tien Tuan Anh Dinh, Ji Wang, Gang Chen, Rui Liu, Beng Chin Ooi, and Kian Lee Tan. BLOCKBENCH: A framework for analyzing private blockchains. *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Part F1277:1085–1100, 2017.

[56] Chethana Dukkipati, Yunpeng Zhang, and Liang Chieh Cheng. Decentralized, BlockChain Based Access Control Framework for the Heterogeneous Internet of Things. *Proceedings of the Third ACM Workshop on Attribute-Based Access Control - ABAC'18*, pages 61–69, 2018.

[57] Hamza Es-Samaali, Aissam Outchakoucht, and Jean Philippe Leroy. A blockchain-based access control for big data. *International Journal of Computer Networks and Communications Security*, 5(7):137, 2017.

[58] Md Sadek Ferdous, Andrea Margheri, Federica Paci, Mu Yang, and Vladimiro Sassone. Decentralised runtime monitoring for access control systems in cloud federations. In *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, pages 2632–2633. IEEE, 2017.

[59] Simone Figorilli, Francesca Antonucci, Corrado Costa, Federico Pallottino, Luciano Raso, Marco Castiglione, Edoardo Pinci, Davide Del Vecchio, Giacomo Colle, Andrea Proto, et al. A blockchain implementation prototype for the electronic open source traceability of wood along the whole supply chain. *Sensors*, 18(9):3133, 2018.

[60] Michael Fröwis and Rainer Böhme. In code we trust?: Measuring the control flow immutability of all smart contracts deployed on ethereum. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 10436 LNCS:357–372, 2017.

[61] Juan Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol: Analysis and applications. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 281–310. Springer, 2015.

[62] Luciano García-Bañuelos, Alexander Ponomarev, Marlon Dumas, and Ingo Weber. Optimized execution of business processes on blockchain. In *International Conference on Business Process Management*, pages 130–146. Springer, 2017.

[63] Sara Ghaemi, Sara Rouhani, Rafael Belchior, Rui S Cruz, Hamzeh Khazaei, and Petr Musilek. A pub-sub architecture to promote blockchain interoperability. *arXiv preprint arXiv:2101.12331*, 2021.

[64] Shelly Grossman, Ittai Abraham, Guy Golan-Gueta, Yan Michalevsky, Noam Rinetzky, Mooly Sagiv, and Yoni Zohar. Online detection of effectively callback free objects with applications to smart contracts. *Proceedings of the ACM on Programming Languages*, 2(POPL):1–28, 2017.

[65] Andreas Grüner, Alexander Mühle, and Christoph Meinel. On the relevance of blockchain in identity management. *arXiv preprint arXiv:1807.08136*, 2018.

[66] Hao Guo, Wanxin Li, Mark Nejad, and Chien-Chung Shen. Access control for electronic health records with hybrid blockchain-edge architecture. *arXiv preprint arXiv:1906.01188*, 2019.

[67] Hao Guo, Ehsan Meamari, and Chien-Chung Shen. Blockchain-inspired event recording system for autonomous vehicles. In *2018 1st IEEE International Conference on Hot Information-Centric Networking (HotICN)*, pages 218–222. IEEE, 2018.

[68] Hao Guo, Ehsan Meamari, and Chien-Chung Shen. Multi-authority attribute-based access control with smart contract. In *Proceedings of the 2019 International Conference on Blockchain Technology*, pages 6–11. ACM, 2019.

[69] Rui Guo, Huixian Shi, Qinglan Zhao, and Dong Zheng. Secure attribute-based signature scheme with multiple authorities for blockchain in electronic health records systems. *IEEE access*, 6:11676–11686, 2018.

[70] Sergio Gusmeroli, Salvatore Piccione, and Domenico Rotondi. A capability-based security approach to manage access control in the internet of things. *Mathematical and Computer Modelling*, 58(5-6):1189–1205, 2013.

[71] Michael Hammer and James Champy. *Reengineering the Corporation: Manifesto for Business Revolution, A*. Zondervan, 2009.

[72] Lei Hang, Israr Ullah, and Do-Hyeun Kim. A secure fish farm platform based on blockchain for agriculture data integrity. *Computers and Electronics in Agriculture*, 170:105251, 2020.

[73] Florian Hawlitschek, Benedikt Notheisen, and Timm Teubner. The limits of trust-free systems: A literature review on blockchain technology and trust in the sharing economy. *Electronic commerce research and applications*, 29:50–63, 2018.

[74] Xiao He, Bohan Qin, Yan Zhu, Xing Chen, and Yi Liu. Spesc: A specification language for smart contracts. In *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*, pages 132–137. IEEE, 2018.

[75] Maurice Herlihy and Eric Koskinen. Transactional boosting: a methodology for highly-concurrent transactional objects. In *Proceedings of the 13th ACM SIGPLAN Symposium on Principles and practice of parallel programming*, pages 207–216. ACM, 2008.

[76] Everett Hildenbrandt, Manasvi Saxena, Nishant Rodrigues, Xiaoran Zhu, Philip Daian, Dwight Guth, Brandon Moore, Daejun Park, Yi Zhang, Andrei Stefanescu, and Grigore Rosu. KEVM: A complete formal semantics of the ethereum virtual machine. *Proceedings - IEEE Computer Security Foundations Symposium*, 2018-July:204–217, 2018.

[77] Yoichi Hirai. Bamboo: a language for morphing smart contracts,, 2018. https://github.com/pirapira/bamboo, (accessed June 30, 2018).

[78] Shuang Hu, Lin Hou, Gongliang Chen, Jian Weng, and Jianhua Li. Reputation-based Distributed Knowledge Sharing System in Blockchain. *Proceedings of the EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services (MobiQuitous)*, pages 476–481, 2018.

[79] Vincent C Hu, David Ferraiolo, Rick Kuhn, Arthur R Friedman, Alan J Lang, Margaret M Cogdell, Adam Schnitzer, Kenneth Sandlin, Robert Miller, Karen Scarfone, et al. Guide to attribute based access control (abac) definition and considerations (draft). *NIST special publication*, 800(162), 2013.

[80] Vincent C Hu, D Richard Kuhn, David F Ferraiolo, and Jeffrey Voas. Attribute-based access control. *Computer*, 48(2):85–88, 2015.

[81] Junqin Huang, Linghe Kong, Hong-Ning Dai, Weiping Ding, Long Cheng, Guihai Chen, Xi Jin, and Peng Zeng. Blockchain based mobile crowd sensing in industrial systems. *IEEE Transactions on Industrial Informatics*, 2020.

[82] Seyoung Huh, Sangrae Cho, and Soohyung Kim. Managing IoT devices using blockchain platform. *International Conference on Advanced Communication Technology, ICACT*, pages 464–467, 2017.

[83] Florian Idelberger, Guido Governatori, Régis Riveret, and Giovanni Sartor. Evaluation of logic-based smart contracts for blockchain systems. In *International Symposium on Rules and Rule Markup Languages for the Semantic Web*, pages 167–183. Springer, 2016.

[84] Mayssa Jemel and Ahmed Serhrouchni. Decentralized access control mechanism with temporal dimension based on blockchain. In *2017 IEEE 14th International Conference on e-Business Engineering (ICEBE)*, pages 177–182. IEEE, 2017.

[85] Herbert Jordan, Bernhard Scholz, and Pavle Subotić. Soufflé: On synthesis of program analyzers. In *International Conference on Computer Aided Verification*, pages 422–430. Springer, 2016.

[86] Jiawen Kang, Rong Yu, Xumin Huang, Maoqiang Wu, Sabita Maharjan, Shengli Xie, and Yan Zhang. Blockchain for secure and efficient data sharing in vehicular edge computing and networks. *IEEE Internet of Things Journal*, 6(3):4660–4670, 2018.

[87] Elena Karafiloski and Anastas Mishev. Blockchain solutions for big data challenges: A literature review. In *IEEE EUROCON 2017-17th International Conference on Smart Technologies*, pages 763–768. IEEE, 2017.

[88] Henry M Kim and Marek Laskowski. Toward an ontology-driven blockchain design for supply-chain provenance. *Intelligent Systems in Accounting, Finance and Management*, 25(1):18–27, 2018.

[89] Tai-Hoon Kim, Rekha Goyat, Mritunjay Kumar Rai, Gulshan Kumar, William J Buchanan, Rahul Saha, and Reji Thomas. A novel trust evaluation process for secure localization using a decentralized blockchain in wireless sensor networks. *IEEE Access*, 7:184133–184144, 2019.

[90] Petar Kochovski, Sandi Gec, Vlado Stankovski, Marko Bajec, and Pavel D Drobintsev. Trust management in a blockchain based fog computing platform with trustless smart oracles. *Future Generation Computer Systems*, 101:747–759, 2019.

[91] Kari Korpela, Jukka Hallikas, and Tomi Dahlberg. Digital supply chain transformation toward blockchain integration. In *proceedings of the 50th Hawaii international conference on system sciences*, 2017.

[92] Kari Korpela, Usva Kuusiholma, Ossi Taipale, and Jukka Hallikas. A framework for exploring digital business ecosystems. In *proceedings of the 46th Hawaii international conference on system sciences*, pages 3838–3847. IEEE, 2013.

[93] Ahmed Kosba, Andrew Miller, Elaine Shi, Zikai Wen, and Charalampos Papamanthou. Hawk: The Blockchain Model of Cryptography and Privacy-Preserving Smart Contracts. *Cryptology ePrint Archive, Report*, pages 1–32, 2015.

[94] Abhiram Kothapalli, Andrew Miller, and Nikita Borisov. SmartCast: An incentive compatible consensus protocol using smart contracts. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 10323 LNCS:536–552, 2017.

[95] Clete A Kushida, Deborah A Nichols, Rik Jadrnicek, Ric Miller, James K Walsh, and Kara Griffin. Strategies for de-identification and anonymization of electronic health record data for use in multicenter research studies. *Medical care*, 50(Suppl):S82, 2012.

[96] Ho-Pun Lam and Guido Governatori. The making of spindle. In *International Workshop on Rules and Rule Markup Languages for the Semantic Web*, pages 315–322. Springer, 2009.

[97] YongJoo Lee and Keon Myung Lee. Blockchain-based rbac for user authentication with anonymity. In *Proceedings of the Conference on Research in Adaptive and Convergent Systems*, pages 289–294. ACM, 2019.

[98] Victoria L Lemieux. Blockchain and distributed ledgers as trusted recordkeeping systems. In *Future Technologies Conference (FTC)*, volume 2017, 2017.

[99] Yue Liu, Qinghua Lu, Shiping Chen, Qiang Qu, Hugo O'Connor, Kim-Kwang Raymond Choo, and He Zhang. Capability-based iot access control using blockchain. *Digital Communications and Networks*, 2020.

[100] Federico Lombardi, Leonardo Aniello, Stefano De Angelis, Andrea Margheri, and Vladimiro Sassone. A blockchain-based infrastructure for reliable and cost-effective IoT-aided smart grids. *IET Conference Publications*, 2018(CP740):1–6, 2018.

[101] Orlenys López-Pintado, Luciano García-Bañuelos, Marlon Dumas, and Ingo Weber. Caterpillar: A blockchain-based business process management system. In *Proceedings of the BPM Demo Track and BPM Dissertation Award co-located with 15th International Conference on Business Process Modeling (BPM 2017), Barcelona, Spain*, 2017.

[102] Yunlong Lu, Xiaohong Huang, Yueyue Dai, Sabita Maharjan, and Yan Zhang. Blockchain and federated learning for privacy-preserved data sharing in industrial iot. *IEEE Transactions on Industrial Informatics*, 16(6):4177–4186, 2019.

[103] Christian Lundkvist, Rouven Heck, Joel Torstensson, Zac Mitton, and Michael Sena. Uport: A platform for self-sovereign identity, 2017. http://blockchainlab.com/pdfuPort_whitepaper_DRAFT20161020.pdf, (accessed Oct 18, 2018).

[104] Loi Luu, Duc-Hiep Chu, Hrishi Olickel, Prateek Saxena, and Aquinas Hobor. Making Smart Contracts Smarter. *Ccs*, 2016.

[105] Mingxin Ma, Guozhen Shi, and Fenghua Li. Privacy-oriented blockchain-based distributed key management architecture for hierarchical access control in the iot scenario. *IEEE Access*, 7:34045–34059, 2019.

[106] Damiano Di Francesco Maesa, Paolo Mori, and Laura Ricci. Blockchain based access control. In *IFIP International Conference on Distributed Applications and Interoperable Systems*, pages 206–220. Springer, 2017.

[107] Damiano Di Francesco Maesa, Paolo Mori, and Laura Ricci. A blockchain based approach for the definition of auditable access control systems. *Computers & Security*, 2019.

[108] James Martin. *Managing the data base environment*. Prentice Hall PTR, 1983.

[109] Laurent Maryline, Kaaniche Nesrine, and Le Christian. A Blockchain based Access Control Scheme. In *Proceedings of the 15th International Joint Conference on e-Business and Telecommunications*, pages 168–176, 2018.

[110] Patrick McCorry, Siamak F. Shahandashti, and Feng Hao. A smart contract for boardroom voting with maximum voter privacy. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 10322 LNCS:357–375, 2017.

[111] Jan Mendling, Ingo Weber, Wil Van Der Aalst, Jan Vom Brocke, Cristina Cabanillas, Florian Daniel, Søren Debois, Claudio Di Ciccio, Marlon Dumas, Schahram Dustdar, et al. Blockchains for business process management-challenges and opportunities. *ACM Transactions on Management Information Systems (TMIS)*, 9(1):4, 2018.

[112] Roberto Metere and Changyu Dong. Automated cryptographic analysis of the pedersen commitment scheme. In *International Conference on Mathematical Methods, Models, and Architectures for Computer Network Security*, pages 275–287. Springer, 2017.

[113] Tomas Mikula and Rune Hylsberg Jacobsen. Identity and access management with blockchain in electronic healthcare records. *Proceedings - 21st Euromicro Conference on Digital System Design, DSD 2018*, pages 699–706, 2018.

[114] Carlos Molina-Jimenez, Ellis Solaiman, Ioannis Sfyrakis, Irene Ng, and Jon Crowcroft. On and off-blockchain enforcement of smart contracts. *arXiv preprint arXiv:1805.00626*, 2018.

[115] Alexander Mühle, Andreas Grüner, Tatiana Gayvoronskaya, and Christoph Meinel. A survey on essential components of a self-sovereign identity. *Computer Science Review*, 30:80–86, 2018.

[116] Marcel Müller, Nadine Ostern, and Michael Rosemann. Silver bullet for all trust issues? blockchain-based trust patterns for collaborative business processes. In *International Conference on Business Process Management*, pages 3–18. Springer, 2020.

[117] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. Technical report, Manubot, 2009.

[118] Yuta Nakamura, Yuanyu Zhang, Masahiro Sasabe, and Shoji Kasahara. Capability-based access control for the internet of things: An ethereum blockchain-based scheme. In *2019 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6. IEEE, 2019.

[119] Gregory S Nelson. Practical implications of sharing data: a primer on data privacy, anonymization, and de-identification. In *SAS Global Forum Proceedings*, pages 1–23, 2015.

[120] Giang-Truong Nguyen and Kyungbaek Kim. A survey about consensus algorithms used in blockchain. *Journal of Information processing systems*, 14(1), 2018.

[121] Ivica Nikolić, Aashish Kolluri, Ilya Sergey, Prateek Saxena, and Aquinas Hobor. Finding the greedy, prodigal, and suicidal contracts at scale. *ACM International Conference Proceeding Series*, pages 653–663, 2018.

[122] Oscar Novo. Blockchain meets iot: An architecture for scalable access management in iot. *IEEE Internet of Things Journal*, 5(2):1184–1195, 2018.

[123] Timothy Nugent, David Upton, and Mihai Cimpoesu. Improving data transparency in clinical trials using blockchain smart contracts. *F1000Research*, 5:1–7, 2016.

[124] Kieron O'hara. Data trusts: Ethics, architecture and governance for trustworthy data stewardship, 2019.

[125] Diego Ongaro and John Ousterhout. In search of an understandable consensus algorithm. In *2014 {USENIX} Annual Technical Conference ({USENIX}{ATC} 14)*, pages 305–319, 2014.

[126] Aafaf Ouaddah, Anas Abou Elkalam, and Abdellah Ait Ouahman. Fairaccess: a new blockchain-based access control framework for the internet of things. *Security and communication networks*, 9(18):5943–5964, 2016.

[127] Aafaf Ouaddah, Anas Abou Elkalam, and Abdellah Ait Ouahman. Towards a novel privacy-preserving access control model based on blockchain technology in iot. In *Europe and MENA Cooperation Advances in Information and Communication Technologies*, pages 523–533. Springer, 2017.

[128] Aissam Outchakoucht, ES Hamza, and Jean Philippe Leroy. Dynamic access control policy based on blockchain and machine learning for the internet of things. *Int. J. Adv. Comput. Sci. Appl*, 8(7):417–424, 2017.

[129] Jordi Paillisse, Jordi Subira, Albert Lopez, Alberto Rodriguez-Natal, Vina Ermagan, Fabio Maino, and Albert Cabellos. Distributed access control with blockchain. *arXiv preprint arXiv:1901.03568*, 2019.

[130] Nikolaos Papoulias. *Remote Debugging and Reflection in Resource Constrained Devices*. PhD thesis, Université des Sciences et Technologie de Lille-Lille I, 2013.

[131] Daejun Park, Yi Zhang, Manasvi Saxena, Philip Daian, and Grigore Roşu. A formal verification tool for ethereum VM bytecode. *ESEC/FSE 2018 - Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 912–915, 2018.

[132] Kevin Peterson, Rammohan Deeduvanu, Pradip Kanjamala, and Kelly Boles. A blockchain-based approach to health information exchange networks. In *Proc. NIST Workshop Blockchain Healthcare*, volume 1, pages 1–10, 2016.

[133] Otto Julio Ahlert Pinno, Andre Ricardo Abed Gregio, and Luis C.E. De Bona. ControlChain: Blockchain as a Central Enabler for Access Control Authorizations in the IoT. *2017 IEEE Global Communications Conference, GLOBECOM 2017 - Proceedings*, 2018-Janua:1–6, 2018.

[134] Serguei Popov. The tangle. *White paper*, 1:3, 2018.

[135] Vahid Pourheidari, Sara Rouhani, and Ralph Deters. A case study of execution of untrusted business process on permissioned blockchain. In *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pages 1588–1594. IEEE, 2018.

[136] Ahmed Raza Rajput, Qianmu Li, Milad Taleby Ahvanooey, and Isma Masood. Eacms: emergency access control management system for personal health record based on blockchain. *IEEE Access*, 7:84304–84317, 2019.

[137] Drummond Reed, Jason Law, and Daniel Hardman. The technical foundations of sovrin a white paper from the sovrin foundation, 2016. https://www.evernym.com/wp-content/uploads/2017/07/The-Technical-Foundations-of-Sovrin.pdf, (accessed Oct 18, 2018).

[138] Paul Rimba, An Binh Tran, Ingo Weber, Mark Staples, Alexander Ponomarev, and Xiwei Xu. Comparing blockchain and cloud services for business process execution. In *2017 IEEE International Conference on Software Architecture (ICSA)*, pages 257–260. IEEE, 2017.

[139] Lillian Røstad and Øystein Nytrø. Personalized access control for a personally controlled health record. In *Proceedings of the 2nd ACM workshop on Computer security architectures*, pages 9–16, 2008.

[140] Sara Rouhani. Security , Performance , and Applications of Smart Contracts : A Systematic Survey. *IEEE Access*, 7:50759–50779, 2019.

[141] Sara Rouhani, Rafael Belchior, Rui S Cruz, and Ralph Deters. Distributed attribute-based access control system using a permissioned blockchain. *World Wide Web*, 2021.

[142] Sara Rouhani, Luke Butterworth, Adam D Simmons, Darryl G Humphery, and Ralph Deters. Medichain tm: a secure decentralized medical data asset management system. In *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pages 1533–1538. IEEE, 2018.

[143] Sara Rouhani and Ralph Deters. Performance analysis of ethereum transactions in private blockchain. In *2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS)*, pages 70–74. IEEE, 2017.

[144] Sara Rouhani and Ralph Deters. Blockchain based access control systems: State of the art and challenges. In *IEEE/WIC/ACM International Conference on Web Intelligence*, WI '19, page 423–428, New York, NY, USA, 2019. Association for Computing Machinery.

[145] Sara Rouhani, Vahid Pourheidari, and Ralph Deters. Physical access control management system based on permissioned blockchain. In *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pages 1078–1083. IEEE, 2018.

[146] Michele Ruta, Floriano Scioscia, Saverio Ieva, Giovanna Capurso, and Eugenio Di Sciascio. Supply chain object discovery with semantic-enhanced blockchain. In *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems*, page 60. ACM, 2017.

[147] Michele Ruta, Floriano Scioscia, Saverio Ieva, Giovanna Capurso, Agnese Pinto, and Eugenio Di Sciascio. A Blockchain Infrastructure for the Semantic Web of Things. *CEUR Workshop Proceedings*, 2161:1DUMMY, 2018.

[148] Pierangela Samarati and Sabrina Capitani de Vimercati. Access control: Policies, models, and mechanisms. In *International School on Foundations of Security Analysis and Design*, pages 137–196. Springer, 2000.

[149] Ravi S Sandhu and Pierangela Samarati. Access control: principle and practice. *IEEE communications magazine*, 32(9):40–48, 1994.

[150] R.S. Sandhu and P. Samarati. Access Control: Principles and Practice. *IEEE Communications Magazine*, 32(9):40–48, 1994.

[151] Franklin Schrans, Susan Eisenbach, and Sophia Drossopoulou. Writing safe smart contracts in flint. In *Conference Companion of the 2nd International Conference on Art, Science, and Engineering of Programming*, pages 218–219. ACM, 2018.

[152] Franklin Schrans, Susan Eisenbach, and Sophia Drossopoulou. Writing safe smart contracts in flint. *ACM International Conference Proceeding Series*, Part F1376(June):218–219, 2018.

[153] Fabian Schuh and Daniel Larimer. Bitshares 2.0: General overview. *accessed June-2017.[Online]. Available: http://docs. bitshares. org/_downloads/bitshares-general. pdf*, 2017.

[154] Pablo Lamela Seijas, Simon J Thompson, and Darryl McAdams. Scripting smart contracts for distributed ledger technology. *IACR Cryptology ePrint Archive*, 2016:1156, 2016.

[155] Ilya Sergey, Amrit Kumar, and Aquinas Hobor. Scilla: a smart contract intermediate-level language. *arXiv preprint arXiv:1801.00687*, 2018.

[156] Ilya Sergey, Amrit Kumar, and Aquinas Hobor. Scilla: a smart contract intermediate-level language. *arXiv preprint arXiv:1801.00687*, 2018.

[157] Zonyin Shae and Jeffrey JP Tsai. On the design of a blockchain platform for clinical trial and precision medicine. In *2017 IEEE 37th international conference on distributed computing systems (ICDCS)*, pages 1972–1980. IEEE, 2017.

[158] Saghar Shah. Block chain voting system, 2016. https://www.economist.com/sites/default/files/northeastern.pdf, (accessed Sep 8, 2018).

[159] Besfort Shala, Ulrich Trick, Armin Lehmann, Bogdan Ghita, and Stavros Shiaeles. Blockchain and trust for secure, end-user-based and decentralized iot service provision. *IEEE Access*, 8:119961–119979, 2020.

[160] Nir Shavit and Dan Touitou. Software transactional memory. *Distributed Computing*, 10(2):99–116, 1997.

[161] Meng Shen, Junxian Duan, Liehuang Zhu, Jie Zhang, Xiaojiang Du, and Mohsen Guizani. Blockchain-based incentives for secure and collaborative data sharing in multiple clouds. *IEEE Journal on Selected Areas in Communications*, 38(6):1229–1241, 2020.

[162] shocard. Shocard with shocoin tokens whitepaper identity management verified using the blockchain, 2017. http://www.lianzhiliao.com/media/whitepapers/ShoCard-Whitepaper_en.pdf, (accessed Oct 18, 2018).

[163] Ajay Kumar Shrestha and Julita Vassileva. User data sharing frameworks: A blockchain-based incentive solution. In *2019 IEEE 10th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, pages 0360–0366. IEEE, 2019.

[164] Sophie Stalla-Bourdillon, Gefion Thuermer, Johanna Walker, Laura Carmichael, and Elena Simperl. Data protection by design: building the foundations of trustworthy data sharing. *Data & Policy*, 2, 2020.

[165] William Stallings, Lawrie Brown, Michael D Bauer, and Arup Kumar Bhattacharjee. *Computer security: principles and practice*. Pearson Education Upper Saddle River, NJ, USA, 2012.

[166] Zhou Su, Yuntao Wang, Qichao Xu, and Ning Zhang. Lvbs: Lightweight vehicular blockchain for secure data sharing in disaster rescue. *IEEE Transactions on Dependable and Secure Computing*, 2020.

[167] Dawei Sun, Guiran Chang, Lina Sun, and Xingwei Wang. Surveying and analyzing security, privacy and trust issues in cloud computing environments. *Procedia Engineering*, 15:2852–2856, 2011.

[168] Nikhil Swamy, Cătălin Hriţcu, Chantal Keller, Aseem Rastogi, Antoine Delignat-Lavaud, Simon Forest, Karthikeyan Bhargavan, Cédric Fournet, Pierre-Yves Strub, Markulf Kohlweiss, et al. Dependent types and multi-monadic effects in f. In *Proceedings of the 43rd annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 256–270, 2016.

[169] Stefan Tai, Jacob Eberhardt, and Markus Klems. Not acid, not base, but salt. In *Proceedings of the 7th International Conference on Cloud Computing and Services Science*, pages 755–764. SCITEPRESS-Science and Technology Publications, Lda, 2017.

[170] Anastasia Theodouli, Stelios Arakliotis, Konstantinos Moschou, Konstantinos Votis, and Dimitrios Tzovaras. On the Design of a Blockchain-Based System to Facilitate Healthcare Data Sharing. *Proceedings - 17th IEEE International Conference on Trust, Security and Privacy in Computing and Communications and 12th IEEE International Conference on Big Data Science and Engineering, Trustcom/BigDataSE 2018*, pages 1374–1379, 2018.

[171] Sergei Tikhomirov, Ekaterina Voskresenskaya, Ivan Ivanitskiy, Ramil Takhaviev, Evgeny Marchenko, and Yaroslav Alexandrov. Smartcheck: Static analysis of ethereum smart contracts. In *Proceedings of the 1st International Workshop on Emerging Trends in Software Engineering for Blockchain*, pages 9–16, 2018.

[172] Petar Tsankov, Andrei Dan, Dana Drachsler-Cohen, Arthur Gervais, Florian Buenzli, and Martin Vechev. Securify: Practical security analysis of smart contracts. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 67–82, 2018.

[173] Marko Vukolić. The quest for scalable blockchain fabric: Proof-of-work vs. bft replication. In *International workshop on open problems in network security*, pages 112–125. Springer, 2015.

[174] Jingzhong Wang, Mengru Li, Yunhua He, Hong Li, Ke Xiao, and Chao Wang. A blockchain based privacy-preserving incentive mechanism in crowdsensing applications. *IEEE Access*, 6:17545–17556, 2018.

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

[176] Wenbo Wang, Dinh Thai Hoang, Zehui Xiong, Dusit Niyato, Ping Wang, Peizhao Hu, and Yonggang Wen. A survey on consensus mechanisms and mining management in blockchain networks. *arXiv preprint arXiv:1805.02707*, 2018.

[177] Hiroki Watanabe, Shigeru Fujimura, Atsushi Nakadaira, Yasuhiko Miyazaki, Akihito Akutsu, and Jay Kishigami. Blockchain contract: Securing a blockchain applied to smart contracts. *2016 IEEE International Conference on Consumer Electronics, ICCE 2016*, pages 467–468, 2016.

[178] Ingo Weber, Xiwei Xu, Régis Riveret, Guido Governatori, Alexander Ponomarev, and Jan Mendling. Untrusted business process monitoring and execution using blockchain. In *International Conference on Business Process Management*, pages 329–347. Springer, 2016.

[179] PengCheng Wei, Dahu Wang, Yu Zhao, Sumarga Kumar Sah Tyagi, and Neeraj Kumar. Blockchain data-based cloud data integrity protection mechanism. *Future Generation Computer Systems*, 102:902–911, 2020.

[180] Stephen A White. Introduction to bpmn. *Ibm Cooperation*, 2(0):0, 2004.

[181] Gavin Wood et al. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151(2014):1–32, 2014.

[182] Bianca Wylie and Sean McDonald. What is a data trust?, 2018.

[183] QI Xia, Emmanuel Boateng Sifah, Kwame Omono Asamoah, Jianbin Gao, Xiaojiang Du, and Mohsen Guizani. Medshare: Trust-less medical data sharing among cloud service providers via blockchain. *IEEE Access*, 5:14757–14767, 2017.

[184] Qi Xia, Emmanuel Boateng Sifah, Abla Smahi, Sandro Amofa, and Xiaosong Zhang. BBDS: Blockchain-based data sharing for electronic medical records in cloud environments. *Information (Switzerland)*, 8(2), 2017.

[185] Ronghua Xu, Yu Chen, Erik Blasch, and Genshe Chen. BlendCAC: A smart contract enabled decentralized capability-based access control mechanism for the IoT. *Computers*, 7(3):1–27, 2018.

[186] Ronghua Xu, Yu Chen, Erik Blasch, and Genshe Chen. Exploration of blockchain-enabled decentralized capability-based access control strategy for space situation awareness. *Optical Engineering*, 58(4):041609, 2019.

[187] Shichang Xuan, Li Zheng, Ilyong Chung, Wei Wang, Dapeng Man, Xiaojiang Du, Wu Yang, and Mohsen Guizani. An incentive mechanism for data sharing based on blockchain with smart contracts. *Computers & Electrical Engineering*, 83:106587, 2020.

[188] Zheng Yan and Silke Holtmanns. Trust modeling and management: from social trust to digital trust. In *Computer security, privacy and politics: current issues, challenges and solutions*, pages 290–323. IGI Global, 2008.

[189] Zhe Yang, Kan Yang, Lei Lei, Kan Zheng, and Victor CM Leung. Blockchain-based decentralized trust management in vehicular networks. *IEEE Internet of Things Journal*, 6(2):1495–1505, 2018.

[190] Affan Yasin and Lin Liu. An Online Identity and Smart Contract Management System. *2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC)*, pages 192–198, 2016.

[191] Lian Yu, Wei Tek Tsai, Guannan Li, Yafe Yao, Chenjian Hu, and Enyan Deng. Smart-Contract Execution with Concurrent Block Building. *Proceedings - 11th IEEE International Symposium on Service-Oriented System Engineering, SOSE 2017*, pages 160–167, 2017.

[192] Eric Yuan and Jin Tong. Attributed based access control (abac) for web services. In *IEEE International Conference on Web Services (ICWS'05)*. IEEE, 2005.

[193] Li Yue, Huang Junqin, Qin Shengzhi, and Wang Ruijin. Big data model of security sharing based on blockchain. In *2017 3rd International Conference on Big Data Computing and Communications (BIGCOM)*, pages 117–121. IEEE, 2017.

[194] Xiao Yue, Huiju Wang, Dawei Jin, Mingqiang Li, and Wei Jiang. Healthcare Data Gateways: Found Healthcare Intelligence on Blockchain with Novel Privacy Risk Control. *Journal of Medical Systems*, 40(10), 2016.

[195] Liudmila Zavolokina, Florian Spychiger, Claudio Tessone, and Gerhard Schwabe. Incentivizing data quality in blockchains for inter-organizational networks–learning from the digital car dossier, 2018.

[196] Fan Zhang, Ethan Cecchetti, Kyle Croman, Ari Juels, and Elaine Shi. Town crier: An authenticated data feed for smart contracts. In *Proceedings of the 2016 aCM sIGSAC conference on computer and communications security*, pages 270–282, 2016.

[197] Peng Zhang, Jules White, Douglas C Schmidt, Gunther Lenz, and S Trent Rosenbloom. Fhirchain: applying blockchain to securely and scalably share clinical data. *Computational and structural biotechnology journal*, 16:267–278, 2018.

[198] Xiaoshuai Zhang and Stefan Poslad. Blockchain support for flexible queries with granular access control to electronic medical records (emr). In *2018 IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE, 2018.

[199] Xiaoshuai Zhang and Stefan Poslad. Blockchain Support for Flexible Queries with Granular Access Control to Electronic Medical Records (EMR). *IEEE International Conference on Communications*, 2018-May:1–6, 2018.

[200] Yuanyu Zhang, Shoji Kasahara, Yulong Shen, Xiaohonh Jiang, and JianxiongWan. Smart Contract-Based Access Control for the Internet of Things. *IEEE Internet of Things Journal*, 6(2):1594–1605, 2019.

[201] Bao-Kun Zheng, Lie-Huang Zhu, Meng Shen, Feng Gao, Chuan Zhang, Yan-Dong Li, and Jing Yang. Scalable and privacy-preserving data sharing based on blockchain. *Journal of Computer Science and Technology*, 33(3):557–567, 2018.

[202] Peilin Zheng, Zibin Zheng, Xiapu Luo, Xiangping Chen, and Xuanzhe Liu. A detailed and real-time performance monitoring framework for blockchain systems. *Proceedings - International Conference on Software Engineering*, pages 134–143, 2018.

[203] Peilin Zheng, Zibin Zheng, Xiapu Luo, Xiangping Chen, and Xuanzhe Liu. A detailed and real-time performance monitoring framework for blockchain systems. In *2018 IEEE/ACM 40th International Conference on Software Engineering: Software Engineering in Practice Track (ICSE-SEIP)*, pages 134–143. IEEE, 2018.

[204] Xiaochen Zheng, Raghava Rao Mukkamala, Ravi Vatrapu, and Joaqun Ordieres-Mere. Blockchain-based personal health data sharing system using cloud storage. In *2018 IEEE 20th International Conference on e-Health Networking, Applications and Services (Healthcom)*, pages 1–6. IEEE, 2018.

[205] Liehuang Zhu, Hui Dong, Meng Shen, and Keke Gai. An incentive mechanism using shapley value for blockchain-based medical data sharing. In *2019 IEEE 5th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing,(HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS)*, pages 113–118. IEEE, 2019.

[206] Yan Zhu, Yao Qin, Guohua Gan, Yang Shuai, and William Cheng-Chung Chu. Tbac: transaction-based access control on blockchain for resource sharing with cryptographically decentralized authorization. In *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*, volume 1, pages 535–544. IEEE, 2018.

[207] Raheel Zubairy. Create a blockchain app for loyalty points with hyperledger fabric ethereum virtual machine, 2018. https://developer.ibm.com/patterns/loyalty-points-fabric-evm/, (accessed Feb, 2019).

[208] Guy Zyskind. Enigma: Decentralized Computation Platform with Guaranteed Privacy. *New Solutions for Cybersecurity*, pages 1–14, 2019.

[209] Guy Zyskind, Oz Nathan, and Alex Pentland. Enigma: Decentralized computation platform with guaranteed privacy. *arXiv preprint arXiv:1506.03471*, 2015.

[210] Guy Zyskind, Oz Nathan, and Alex 'Sandy' Pentland. Decentralizing Privacy: Using Blockchain to Protect Personal Data. In *IEEE Security and Privacy Workshops*, pages 180–184, 2015.

Appendix

Licence to republish

For the chapter 6, the licence is attached in the bellowing. For the other papers all are published in IEEE and "The IEEE does not require individuals working on a thesis to obtain a formal reuse license" (see the attached form).

IEEE
Requesting permission to reuse content from an IEEE publication

Security, Performance, and Applications of Smart Contracts: A Systematic Survey

Author: Sara Rouhani
Publication: IEEE Access
Publisher: IEEE
Date: 2019

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

BACK CLOSE WINDOW

SPRINGER NATURE LICENSE
TERMS AND CONDITIONS

Mar 25, 2021

This Agreement between Ms. Sara Rouhani ("You") and Springer Nature ("Springer Nature") consists of your license details and the terms and conditions provided by Springer Nature and Copyright Clearance Center.

License Number	5036110243527
License date	Mar 25, 2021
Licensed Content Publisher	Springer Nature
Licensed Content Publication	World Wide Web
Licensed Content Title	Distributed attribute-based access control system using permissioned blockchain
Licensed Content Author	Sara Rouhani et al
Licensed Content Date	Mar 23, 2021
Type of Use	Thesis/Dissertation
Requestor type	academic/university or research institute
Format	print and electronic
Portion	full article/chapter
Will you be translating?	no
Circulation/distribution	1 - 29
Author of this Springer Nature content	yes

Title	Distributed attribute-based access control system using permissioned blockchain
Institution name	University of Saskatchewan
Expected presentation date	Apr 2021
Requestor Location	Ms. Sara Rouhani 2521 Ewart Ave Saskatoon, SK S7J 1Y7 Canada Attn: University of Alberta
Total	0.00 CAD

Terms and Conditions

Springer Nature Customer Service Centre GmbH
Terms and Conditions

This agreement sets out the terms and conditions of the licence (the **Licence**) between you and **Springer Nature Customer Service Centre GmbH** (the **Licensor**). By clicking 'accept' and completing the transaction for the material (**Licensed Material**), you also confirm your acceptance of these terms and conditions.

1. Grant of License

1. 1. The Licensor grants you a personal, non-exclusive, non-transferable, world-wide licence to reproduce the Licensed Material for the purpose specified in your order only. Licences are granted for the specific use requested in the order and for no other use, subject to the conditions below.

1. 2. The Licensor warrants that it has, to the best of its knowledge, the rights to license reuse of the Licensed Material. However, you should ensure that the material you are requesting is original to the Licensor and does not carry the copyright of another entity (as credited in the published version).

1. 3. If the credit line on any part of the material you have requested indicates that it was reprinted or adapted with permission from another source, then you should also seek permission from that source to reuse the material.

2. Scope of Licence

2. 1. You may only use the Licensed Content in the manner and to the extent permitted by these Ts&Cs and any applicable laws.

2. 2. A separate licence may be required for any additional use of the Licensed Material, e.g. where a licence has been purchased for print only use, separate permission must be obtained for electronic re-use. Similarly, a licence is only valid in the language selected and does not apply for editions in other languages unless additional translation rights have been granted separately in the licence. Any content

owned by third parties are expressly excluded from the licence.

2. 3. Similarly, rights for additional components such as custom editions and derivatives require additional permission and may be subject to an additional fee. Please apply to Journalpermissions@springernature.com/bookpermissions@springernature.com for these rights.

2. 4. Where permission has been granted **free of charge** for material in print, permission may also be granted for any electronic version of that work, provided that the material is incidental to your work as a whole and that the electronic version is essentially equivalent to, or substitutes for, the print version.

2. 5. An alternative scope of licence may apply to signatories of the STM Permissions Guidelines, as amended from time to time.

3. Duration of Licence

3. 1. A licence for is valid from the date of purchase ('Licence Date') at the end of the relevant period in the below table:

Scope of Licence	Duration of Licence
Post on a website	12 months
Presentations	12 months
Books and journals	Lifetime of the edition in the language purchased

4. Acknowledgement

4. 1. The Licensor's permission must be acknowledged next to the Licenced Material in print. In electronic form, this acknowledgement must be visible at the same time as the figures/tables/illustrations or abstract, and must be hyperlinked to the journal/book's homepage. Our required acknowledgement format is in the Appendix below.

5. Restrictions on use

5. 1. Use of the Licensed Material may be permitted for incidental promotional use and minor editing privileges e.g. minor adaptations of single figures, changes of format, colour and/or style where the adaptation is credited as set out in Appendix 1 below. Any other changes including but not limited to, cropping, adapting, omitting material that affect the meaning, intention or moral rights of the author are strictly prohibited.

5. 2. You must not use any Licensed Material as part of any design or trademark.

5. 3. Licensed Material may be used in Open Access Publications (OAP) before publication by Springer Nature, but any Licensed Material must be removed from OAP sites prior to final publication.

6. Ownership of Rights

6. 1. Licensed Material remains the property of either Licensor or the relevant third party and any rights not explicitly granted herein are expressly reserved.

7. Warranty

IN NO EVENT SHALL LICENSOR BE LIABLE TO YOU OR ANY OTHER PARTY OR ANY OTHER PERSON OR FOR ANY SPECIAL, CONSEQUENTIAL, INCIDENTAL OR INDIRECT DAMAGES, HOWEVER CAUSED, ARISING OUT OF OR IN CONNECTION WITH THE DOWNLOADING, VIEWING OR USE OF THE MATERIALS REGARDLESS OF THE FORM OF ACTION, WHETHER FOR BREACH OF CONTRACT, BREACH OF WARRANTY, TORT, NEGLIGENCE, INFRINGEMENT OR OTHERWISE (INCLUDING, WITHOUT LIMITATION, DAMAGES BASED ON LOSS OF PROFITS, DATA, FILES, USE, BUSINESS OPPORTUNITY OR CLAIMS OF THIRD PARTIES), AND
WHETHER OR NOT THE PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. THIS LIMITATION SHALL APPLY NOTWITHSTANDING ANY FAILURE OF ESSENTIAL PURPOSE OF ANY LIMITED REMEDY PROVIDED HEREIN.

8. Limitations

8. 1. ***BOOKS ONLY:*** Where **'reuse in a dissertation/thesis'** has been selected the following terms apply: Print rights of the final author's accepted manuscript (for clarity, NOT the published version) for up to 100 copies, electronic rights for use only on a personal website or institutional repository as defined by the Sherpa guideline (www.sherpa.ac.uk/romeo/).

8. 2. For content reuse requests that qualify for permission under the STM Permissions Guidelines, which may be updated from time to time, the STM Permissions Guidelines supersede the terms and conditions contained in this licence.

9. Termination and Cancellation

9. 1. Licences will expire after the period shown in Clause 3 (above).

9. 2. Licensee reserves the right to terminate the Licence in the event that payment is not received in full or if there has been a breach of this agreement by you.

Appendix 1 — Acknowledgements:

For Journal Content:
Reprinted by permission from [**the Licensor**]: [**Journal Publisher** (e.g. Nature/Springer/Palgrave)] [**JOURNAL NAME**] [**REFERENCE CITATION** (Article name, Author(s) Name), [**COPYRIGHT**] (year of publication)

For **Advance Online Publication papers:**
Reprinted by permission from [**the Licensor**]: [**Journal Publisher** (e.g. Nature/Springer/Palgrave)] [**JOURNAL NAME**] [**REFERENCE CITATION** (Article name, Author(s) Name), [**COPYRIGHT**] (year of publication), advance online publication, day month year (doi: 10.1038/sj.[JOURNAL ACRONYM].)

For Adaptations/Translations:
Adapted/Translated by permission from [**the Licensor**]: [**Journal Publisher** (e.g. Nature/Springer/Palgrave)] [**JOURNAL NAME**] [**REFERENCE CITATION** (Article name, Author(s) Name), [**COPYRIGHT**] (year of publication)

Note: For any republication from the British Journal of Cancer, the following credit line style applies:

Reprinted/adapted/translated by permission from [**the Licensor**]: on behalf of Cancer Research UK: : [**Journal Publisher** (e.g. Nature/Springer/Palgrave)] [**JOURNAL NAME**] [**REFERENCE CITATION** (Article name, Author(s) Name), [**COPYRIGHT]** (year of publication)

For **Advance Online Publication** papers:
Reprinted by permission from The [**the Licensor**]: on behalf of Cancer Research UK: [**Journal Publisher** (e.g. Nature/Springer/Palgrave)] [**JOURNAL NAME**] [**REFERENCE CITATION** (Article name, Author(s) Name), [**COPYRIGHT**] (year of publication), advance online publication, day month year (doi: 10.1038/sj. [JOURNAL ACRONYM])

For Book content:
Reprinted/adapted by permission from [**the Licensor**]: [**Book Publisher** (e.g. Palgrave Macmillan, Springer etc) [**Book Title**] by [**Book author**(s)] [**COPYRIGHT**] (year of publication)

Other Conditions:

Version 1.3

Questions? customercare@copyright.com or +1-855-239-3415 (toll free in the US) or +1-978-646-2777.