Combining Multiple ID's, Attributes, and Policies to Provide Secure Access Control within Hyperledger Fabric Blockchain Networks

by

Daan Gordijn
Supervisor(s): Kaitai Liang, Roland Kromes
EEMCS, Delft University of Technology, The Netherlands

22-6-2022

A
Dissertation
Submitted to EEMCS Faculty
of Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering

Abstract

Blockchain technologies allow users to securely store and trace their data on a fully decentralized system, and have the potential to make a huge impact on many industries. While traditional, permissionless blockchains such as Bitcoin, Ethereum, and Cardano are very popular, they are currently unable to provide trust and privacy on the network. To solve these issues, many new, permissioned blockchain technologies have been implemented, including Hyperledger Fabric. Although Hyperledger Fabric has proven to be highly successful in providing trust and privacy through the use of identities, channels, and private data collections, one of its major drawbacks is the lack of a flexible and scalable access control system. Currently, access control decisions have to be built into each smart contract individually, which can cause many vulnerabilities and prevent access policies to be updated dynamically.

This research aims to answer the research question "How can secure access control in Hyperledger Fabric be guaranteed by combining multiple ID's, attributes, and policies with the components that regulate access control?". To answer this question, the access control system currently used by Hyperledger Fabric is first completely analyzed. Next, a new implementation is proposed that builds upon the existing solution but provides users and developers with easier ways to make access control decisions based on combinations of multiple ID's, attributes, and policies. This solution is then implemented using the smart contract technology of Hyperledger Fabric, which allows it to easily be deployed to existing Hyperledger Fabric networks. Finally, the performance impact of this proposed implementation is analyzed, and new areas of research are proposed that can potentially be explored in future papers.

At the end of this research, it was concluded that it is possible to combine multiple ID's, attributes, and policies with the help of Hyperledger Fabric's smart contract technology. Furthermore, it could be seen that the performance impact for real-world applications is negligible compared to the insecure case of always providing access to a resource without performing access control.

1 Introduction

Ever since the anonymous Satoshi Nakamoto published his Bitcoin white paper [1] in 2008, blockchain has become one of the most disruptive technologies in the computer science industry. In recent years, many other innovative blockchain technologies have been developed [2], which are becoming increasingly more popular.

While Bitcoin was created to provide a digital alternative to traditional, bank-controlled currencies [3], many of these newer blockchain technologies are designed to provide a platform for building and deploying decentralized applications through the use of smart contracts[1]. By implementing their business logic within these smart contracts, decentralized applications can automatically execute any transaction without human intervention, making them completely independent and decentralized [5]. Due to the many benefits of decentralized applications [6], the adoption of blockchain technologies has recently expanded to many non-financial applications such as "healthcare, supply chain management, market monitoring, smart energy, and copyright protection" [7].

Most of these traditional blockchain technologies, such as Bitcoin, Ethereum, and Cardano, are so-called "permissionless" blockchain technologies. This type of blockchain technology, however, has many privacy issues when it is being used in the context of enterprise-level applications, as described in [8]. Many alternative, so-called "permissioned" blockchain technologies have been proposed to solve the issues, the most promising of which is Hyperledger Fabric [9]. Through the use of innovative concepts such as channels, policies, identities, and Membership Service Providers, Hyperledger Fabric can determine the identity of participants, perform access control based on these identities, and ensure the privacy of transactions and smart contracts.

As with many technologies, the increase in popularity of blockchain technologies also drives an increase in security threats and attacks. One of the major issues that many blockchain technologies, including Hyperledger Fabric, currently have is providing secure access control to the distributed ledger and smart contracts. Hyperledger Fabric partially addresses this issue by only granting network access upon submission of a valid X.509 certificate [10], issued and approved by a trusted Certificate Authority. However, this type of ID-based access control is not scalable for larger organizations. This paper will therefore investigate how secure access control in Hyperledger Fabric can be improved, in particular by looking into solutions that can combine these ID's with attributes and policies.

The research question that has been formulated for this study is "How can secure access control in Hyperledger Fabric be guaranteed by combining multiple ID's, attributes, and policies with the components that regulate access control?". In order to systematically answer this research question, the following sub-questions will be addressed:

- What is Hyperledger Fabric?
- What is secure access control in the context of Hyperledger Fabric?
- What are the components that regulate access control within Hyperledger Fabric?
- How are the components for access control currently interacting within Hyperledger Fabric?
- How can multiple ID's, attributes, and policies be combined within Hyperledger Fabric?
- What is the performance impact of ID-, attribute-, and policy-based access control within Hyperledger Fabric?

[1] A digital contract written into code that is stored and automatically executed on the nodes of a distributed blockchain network [4]

This paper is structured in the following manner. First, Section 2 will provide a summary of the most relevant work that currently exists in literature. Next, Section 3 will provide an overview of the methodology that has been applied during this research. Then, Sections 4 and 5 will explain the proposed system model that has been implemented and analyzed as part of this research. Subsequently, Section 6 will provide an overview and analysis of the results that have been obtained during the research, while Section 7 will provide a brief discussion of the research as a whole. Finally, Section 8 will present the main conclusions and possible improvements for future research, while Section 9 will provide a reflection on the ethical aspects and reproducibility of this research.

2 Related Work

Research into secure access control in various blockchain technologies, including Hyperledger Fabric, has been conducted in multiple papers. Many of these studies are performed in the context of exploring the integration of blockchain technologies with the Internet of Things (IoT), as blockchain is currently seen as the most promising technique for providing secure access control to IoT devices [11].

In [12], a summary of the major problems of modern access control systems is presented, together with an explanation of how these problems can potentially be solved using blockchain technologies. In addition, this paper provides an overview of existing access control studies and describes the current challenges of blockchain-based access control.

In [11], an attribute-based access control scheme for Internet of Things devices is proposed by employing blockchain technology to keep track of the distribution of the attributes. In [13], a different scheme is proposed that is built upon various smart contracts and so-called "functional modules", which are jointly responsible for managing attribute information and making access control decisions. In [14], yet another access control scheme is proposed which is implemented and deployed using the smart contract technology of the Ethereum blockchain network.

While the papers discussed so sofar describe general blockchain-based access control systems, other papers make specific use of the Hyperledger Fabric blockchain technology. First, [15], [16], and [17] explore basic access control scenarios for IoT devices in Hyperledger Fabric. Next, [18] combines the Hyperledger Fabric blockchain technology with the InterPlanetary File System (IPFS) [19], allowing IoT devices to easily store documents on a distributed file system and store the hashes of these documents on the blockchain ledger. Finally, [20] proposes a multi-layered and multi-model access control system in the context of an agricultural supply chain system that runs on Hyperledger Fabric.

While research into secure access control in Hyperledger Fabric and other blockchain technologies certainly exists, no study into combining multiple ID's, attributes, and policies during the decision-making process has been conducted. To fill in this gap, this paper will propose a new access control scheme that combines these ID's, attributes, and policies within a single smart contract deployed to a Hyperledger Fabric network. For consistency, this research paper will consider the scenario where an IoT device wants to store a document on IPFS and subsequently save the returned document hash on the blockchain network, as also used in [18].

3 Methodology

The research conducted for this paper consisted of two main parts. First, a *literature research* was conducted to study the existing solutions for secure access control, both within and outside the context of Hyperledger Fabric. Second, a new design to provide secure access control in Hyperledger Fabric was proposed, which was then implemented and analyzed during the *implementation* part using a smart contract.

3.1 Literature Research

During the literature research, Google Scholar was used to find existing literature about secure access control in Hyperledger Fabric and other blockchain technologies. The exact search queries that have been used during this Google Scholar search can be found in Appendix 1. In addition to Google Scholar, the snowball method and citation searching method [21] have been applied to retrieve additional literature that was not listed in the initial search results. A full summary of the most important literature gathered during the literature research phase can be found in Section 2.

3.2 Implementation

Using the existing literature from the previous phase, a new design was proposed to provide secure access control in Hyperledger Fabric. As stated in the research question, this design had to combine multiple ID's, attributes, and policies in the decision-making process. Subsequently, during the implementation phase, the design was implemented using a smart contract and deployed to a local Hyperledger Fabric test network, which was set up using the official tutorial [22].

Hyperledger Fabric currently supports three programming languages for the development of smart contracts and client applications: Go, Java, and NodeJS [23]. For each language, several SDK's are available [24] that help make the implementation of smart contracts and client applications easier. For this particular research project, NodeJS with TypeScript has been selected as the toolchain for the implementation phase, as this language is very easy to learn and understand.

The complete repository that contains a basic test network together with the smart contracts and sample applications that have been implemented during this research project is available on GitHub[2]. The `README` stored in this repository also includes a small tutorial, as well as a complete overview of the required tools and their recommended versions.

4 Current Implementation

This section discusses the current approach to secure access control in Hyperledger Fabric. This section begins with a brief introduction to Hyperledger Fabric and secure access control in general, and subsequently discusses the main components and methodologies that Hyperledger Fabric currently uses to provide secure access control.

[2] https://github.com/daangordijn/Fabric-Access-Control

4.1 Hyperledger Fabric

Hyperledger Fabric is an "open-source enterprise-grade permissioned distributed ledger technology (DLT) platform, designed for use in enterprise contexts" [23]. While many well-established blockchain platforms such as Bitcoin and Ethereum are currently being modified to be used in enterprise-grade applications, Hyperledger Fabric has been built around enterprise applications from the beginning. First, Hyperledger Fabric is highly modular, which allows core parts of the blockchain network to be customized. Second, Hyperledger Fabric has support for writing smart contracts in general-purpose languages, including Go, Java, and NodeJS, while most other blockchain technologies require developers to learn new languages, such as Vyper or Solidity in the case of Ethereum [25]. Finally, Hyperledger Fabric is permissioned, which means that the identity of all participants of the network is known and can therefore be verified using access control systems, allowing organizations to establish trust.

Figure 1 shows the main architecture of a Hyperledger Fabric network that consists of four organizations, R1, R2, R3, and R4. Each organization operates its own Certificate Authority (CA) node, as shown in the top left corner. These nodes are used by the organizations to issue X.509 certificates [10], which are required when peer nodes, orderer nodes, or client applications want to connect to the blockchain network. In addition, organizations R1, R2, and R3 each have a client application, A1, A2, and A3, respectively, which can interact with the shared ledger through a smart contract.

Within Hyperledger Fabric, a network can be split into multiple so-called "channels", each maintaining a separate distributed ledger. Channels are used to ensure the privacy of smart contracts and ledger data since organizations can be granted or denied access to one or more channels on an individual basis. In Figure 1, organizations R1 and R2 are allowed to transact on channel 1, while organizations R2 and R3 are allowed to transact on channel 2. Each channel maintains its own "channel configuration", indicated by CC1 and CC2, respectively, while the overall blockchain network also maintains a "network configuration", indicated by NC1.

Next, as shown in Figure 1, organizations R1, R2, and R3 each operate one peer node, P1, P2, and P3, respectively. These peer nodes are jointly responsible for managing and endorsing transaction proposals. To do this, these peers have to store a copy of all smart contracts and their associated ledgers from all channels they are connected to. As can be seen, peer nodes P1 and P2 store a copy of smart contract S1 and its associated ledger L1, while peer nodes P2 and P3 store a copy of smart contract S2 and its associated ledger L2. In addition to these peer nodes, organization R4 manages an orderer node, O4, which is responsible for ordering the transaction proposals when they are endorsed by the peers.

Finally, as shown in Figure 1, each node in the network maintains a local Membership Service Provider (MSP). These service providers store all X.509 certificates that have been issued by the Certificate Authorities of their corresponding organizations, which are then used by network nodes to map X.509 identities to internal roles. Together with the Certificate Authorities, these providers are therefore responsible for providing the initial layer of identity-based access control.

While the discussion above is nowhere near complete, it provides sufficient information to understand the rest of this paper. Readers who want more information about the Hyperledger Fabric architecture can refer to the official Hyperledger Fabric documentation, available at [9].

4.2 Secure Access Control

Access control is "a security technique that regulates who or what can view or use resources in a computing environment" [27]. Different types of access control exist, including Identity-Based Access Control (IBAC), Role-Based Access Control (RBAC), and Attribute-Based Access Control (ABAC) [28]. While older, established blockchain technologies such as Bitcoin and Ethereum are non-permissioned and therefore do not implement these types of access control systems, Hyperledger Fabric is a permissioned blockchain technology, which enforces it to perform access control.

Currently, Hyperledger Fabric employs multiple layers of access control to provide security and privacy within the blockchain network. First, at the most basic level, Hyperledger Fabric uses a simple identity-based access control system, which prevents unauthorized entities from accessing anything on the blockchain network. This layer is explained in more detail in Sections 4.3 and 4.4 since the purpose of this paper is to extend this simple system to a more complex attribute-based access control system. Second, at an organizational level, Hyperledger Fabric can restrict access to smart contracts and the ledger through the use of channels, as described in Section 4.1. By only granting individual organizations access to the minimal required subset of channels, the privacy of smart contracts and ledger states can be preserved.

4.3 Certificate Authorities (CAs)

A Certificate Authority is an "organization that acts to validate the identities of entities (...) and bind them to cryptographic keys through the issuance of electronic documents known as digital certificates" [29]. Hyperledger Fabric provides a special implementation, called the "Fabric Certificate Authority" or "Fabric CA" in short, which can be used to create and sign these digital certificates using the international X.509 standard [30]. Fabric CA consists of both a client-side and server-side command line interface (CLI), called `fabric-ca-client` and `fabric-ca-server`, respectively. Fabric CA provides many features including "registration of identities, issuance of enrollment certificates, and certificate renewal and revocation" [31].

When an administrator wants to enroll a new identity, Fabric CA will generate a key-value pair that consists of a private key and a public key. Together with the parameters provided by the administrator, a Certificate Signing Request (CSR) will be created, which is then processed by Fabric CA.

In Section 4.5, this process of registering and enrolling a new identity with the Fabric CA server is visualized. This section will also describe a new command line interface (CLI) that has been implemented as part of this study and makes the creation of new identities much easier.

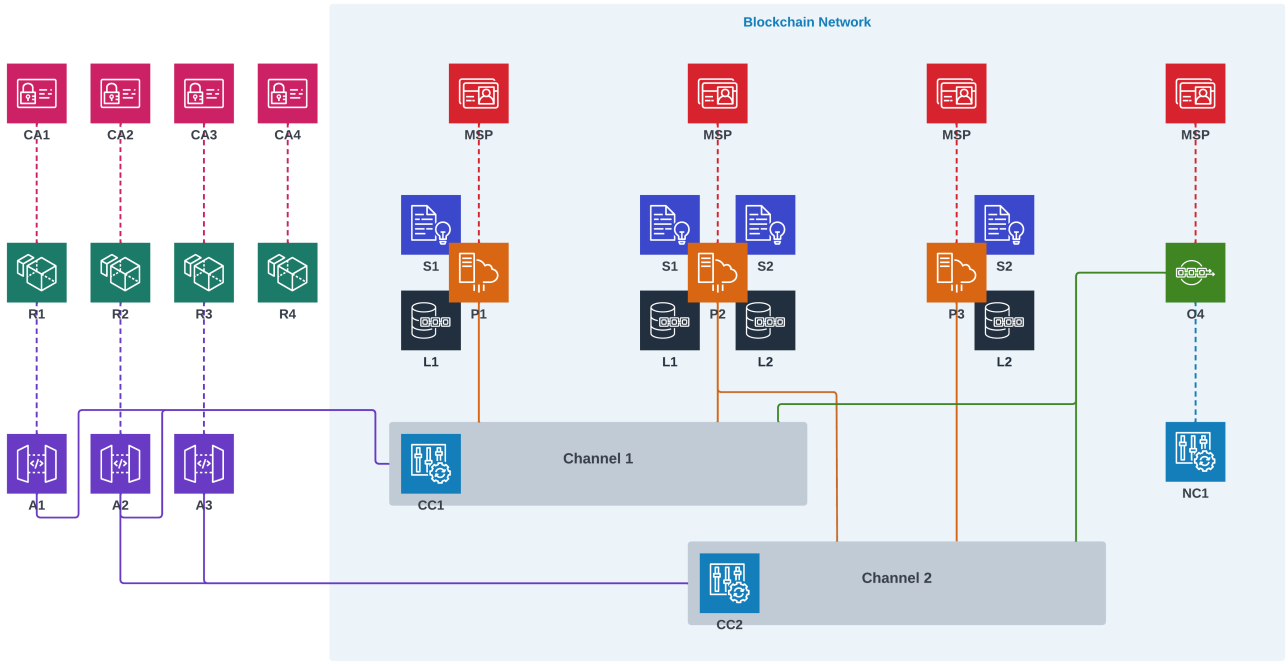

Figure 1: Example of the Hyperledger Fabric architecture. The blockchain network is maintained by four organizations, R1, R2, R3, and R4, each of which operates one or more nodes within the network. Source: Image adapted from [26].

4.4 Membership Service Providers (MSPs)

A Membership Service Provider is a component within Hyperledger Fabric that can be used by participants of the blockchain network to prove their identity to other participants of this network. When a user wants to start interacting with a Hyperledger Fabric blockchain network, it needs to create a key pair, which consists of a public key and a private key, which is needed to prove its identity to the rest of the network. Next, this public key must be included in a Certificate Signing Request (CSR), which is then submitted to a Certificate Authority and used to issue a new X.509 certificate. While X.509 certificates, including public keys, can be shared publicly, private keys must always be kept secret to comply with the principles of Public-Key Infrastructure (PKI) [32].

When a participant of the blockchain network now wants to submit a transaction, it needs to create a transaction proposal and sign this proposal using its private key. All nodes on the blockchain network are then able to verify this transaction proposal using the public X.509 certificate of this participant since it is stored inside the Membership Service Providers. Because of this, Membership Service Providers can establish trust on the permissioned blockchain network, without the need of sharing private keys.

4.5 Generating Certificates

In Figure 2, a simplified version of the process of generating X.509 certificates using Fabric CA is visualized. As can be seen, the Fabric CA Client has to invoke the Fabric CA Server using two commands, `fabric-ca-client register` and `fabric-ca-client enroll` [31]. By doing this, the server will generate a private key, a public key, and a corresponding signed X.509 certificate. This certificate is then automatically stored in the Membership Service Providers that are located on various nodes inside the blockchain network.

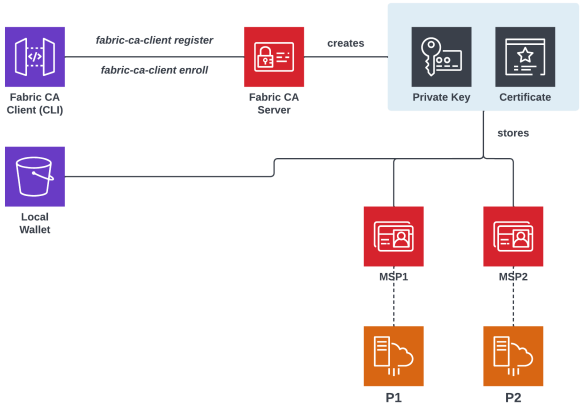

Figure 2: Current process of enrolling a new identity within a Hyperledger Fabric network. The `fabric-ca-client` CLI is used to run the `register` and `enroll` commands, respectively. Then, the resulting X.509 certificate is stored on a set of peer nodes, while both the X.509 certificate and the corresponding private key are stored in the user's local file system wallet.

While this process of generating X.509 certificates for Hyperledger Fabric is not overly complicated, it can become cumbersome to run multiple commands with many different

flags to just create one certificate. Therefore, as part of this research paper, a wrapper around the `fabric-ca-client` was created. This tool, called `certgen`, is publicly available in the GitHub repository[2], together with a small tutorial on how to interact with it. The `certgen` tool internally uses the `fabric-ca-client` commands and has the advantage that it can automatically populate a local file system wallet with the correct files which are required to connect a client application to the blockchain network. In addition, since this tool is highly interactive, it makes it much easier for administrators to add attributes to the certificate. More about the importance of setting attributes within X.509 certificates will be explained in Section 5.

5 Proposed Implementation

This section discusses the proposed implementation that improves the current implementation of secure access control in Hyperledger Fabric, introduced in Section 4. This section begins with a brief discussion of how to independently combine multiple ID's, attributes, and policies, and subsequently presents the final design incorporating these components.

5.1 Combining Attributes

In Hyperledger Fabric, every X.509 certificate issued by Fabric CA [31] can have attributes. These attributes can be used during access control to determine whether a client should be given access, or not. To allow for more complex access control decisions, multiple attributes can be combined into so-called "policies", which are visualized in Figure 3.

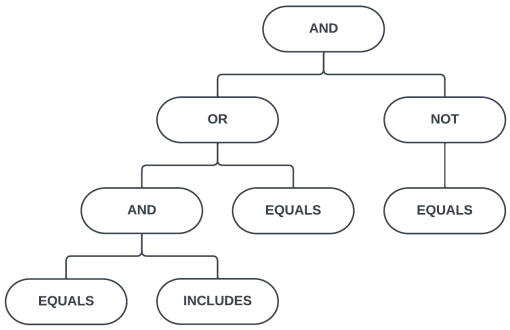

Figure 3: Combining multiple attributes. The `EQUALS` and `INCLUDES` operators validate whether a specified attribute equals or includes a certain value, respectively. The `AND`, `OR`, and `NOT` boolean operators can be then be applied to combine or negate these individual attribute checks, allowing the client to create complex policies.

For this study, the following boolean operators have been selected that can be used for building access control policies:

- `EQUALS`: Checks whether an attribute is present on the certificate, and whether it is equal to the provided value.
- `INCLUDES`: Checks whether an attribute is present on the certificate, and whether it includes the provided value. This operator can be used when the specified attribute on the certificate has a comma-separated list of strings as its value, which must include a particular value.
- `AND`: Logical operator that combines two or more operator trees. This operator returns `true` if and only if all operator trees combined by this operator evaluate to `true`, and returns `false` otherwise.
- `OR`: Logical operator that combines two or more operator trees. This operator returns `true` if and only if at least one of the operator trees combined by this operator evaluates to `true`, and returns `false` otherwise.
- `NOT`: Logical operator that negates the output of another the given tree. This operator returns `true` if and only if the operator tree provided to this operator evaluates to `false`, and returns `false` otherwise.

Together, these operators can build complex policies that can later be evaluated to determine whether a client has access to a resource on the blockchain network, or not.

5.2 Combining Policies

As described in the previous subsection, an access policy is a rule that enforces an X.509 certificate to possess a particular combination of attributes and values. These access policies can be used in Hyperledger Fabric to verify whether an entity invoking a smart contract has sufficient permissions to invoke the endpoint. Figure 4 shows a simplified example of a client invoking three different operations on a smart contract: reading an asset, updating the asset, and deleting the asset.

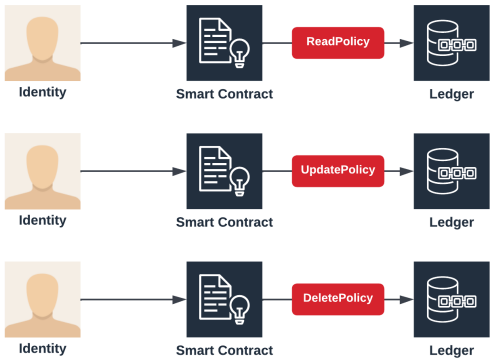

Figure 4: Combining multiple policies. Each smart contract has a different purpose and might need different policies for different operations. Multiple policies can be defined in a single smart contract, and depending on the operation requested by the client, the correct validation policy will be selected and used for access control.

As can be seen in the image, the invoked smart contract has a different access policy for each of the three supported operations. For example, a client might be able to satisfy the `ReadPolicy` with its X.509 certificate, but might not be able to satisfy the `UpdatePolicy` and `DeletePolicy`. Therefore, this client will only be allowed to read the asset and will be denied access when it tries to update or delete the asset.

5.3 Combining ID's

In Hyperledger Fabric, IDs are composed of X.509 certificates [10], issued by Certificate Authorities and managed

by Membership Service Providers. Research into combining multiple such X.509 certificates has not been published to the date of writing. In fact, X.509 certificates cannot be combined by a simple merge, since the X.509 standard [30] does not allow this. Therefore, for this study, alternative ways of combining multiple X.509 certificates had to be found.

The solution proposed in this study can integrate one X.509 certificate, referred to as the "parent", into another X.509 certificate. The process by which this integration can be realized is visualized in Figure 5 and described below.

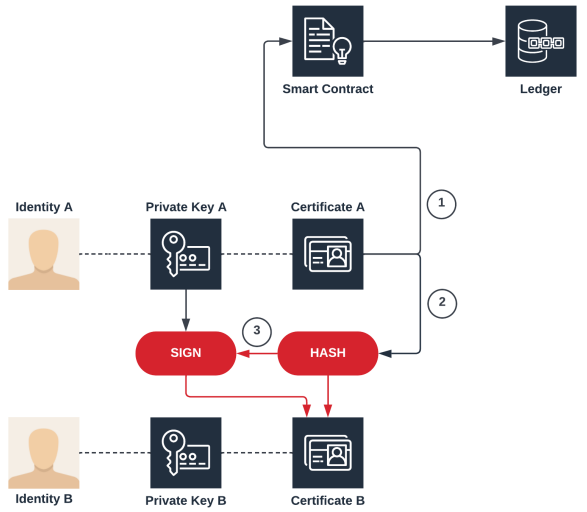

Figure 5: Combining multiple ID's. First, the X.509 certificate of identity A is hashed. Next, this hash is signed with the private key of identity A. Finally, these two values, `hash(certificate)` and `sign(hash(certificate))`, are added to the X.509 certificate of identity B as custom attributes.

- First, the client invokes a special smart contract using certificate A. This smart contract then extracts the certificate from the request, and subsequently stores it into a hashmap on the distributed blockchain ledger;
- Second, the client creates the SHA-256 hash of certificate A, and stores this value in certificate B by setting the `hfa.ParentHash` attribute;
- Third, the client signs the obtained SHA-256 hash using private key A, and stores this value in certificate B by setting the `hfa.ParentSignature` attribute.

Whenever a client now invokes a smart contract on the blockchain network using identity B, this smart contract can verify that this client also owns identity A, since it needed access to private key A in step (3) to calculate the `hfa.ParentSignature` attribute. If the client would not have access to this private key, the signature provided in this attribute cannot be valid. Since certificate A was previously stored on the ledger in step (1), the invoked smart contract has access to the public key of identity A, and could therefore easily establish that the provided signature was forged, thus denying access to the network.

Having established that the client invoking the smart contract with identity B also owns identity A, the smart contract can retrieve the certificate of identity A from the hashmap stored on the distributed ledger, and use it to make access control decisions. The proposed smart contract has been implemented and made available in the GitHub repository[2]. This implementation currently supports one parent certificate to be set in the `hfa.ParentHash` and `hfa.ParentSignature` attributes, although it can easily be extended to support multiple parents or recursive ancestor lookups in the future.

5.4 Final System Design

In the previous subsections, the proposed methods of combining multiple ID's, attributes, and policies have been discussed on an individual basis. This subsection will explain how these three concepts will fit together, and how this combined design has been implemented using Hyperledger Fabric. Figure 6 shows a simplified version of the final system architecture[3].

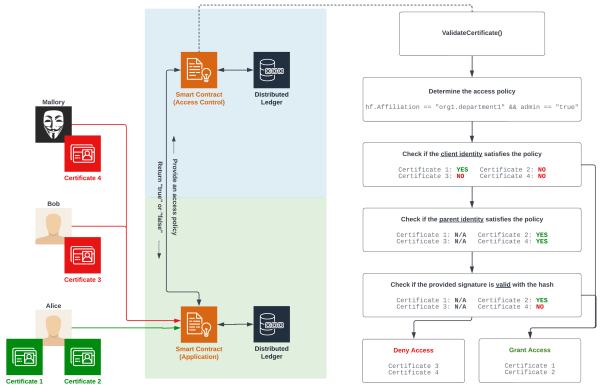

Figure 6: Final system design[3], combining all discussed concepts. Certificate 1 is granted access to the resource since it satisfies the defined access policy. Certificate 2 is granted access to the resource since it contains the `hfa.ParentHash` and `hfa.ParentSignature` attributes, which connects it to certificate 1. Certificate 3 is denied access since it does not satisfy the access policy, while certificate 4 is denied access since it contains an invalid hash signature.

The final system design consists of four main components, which will be described below.

Fabric CA Server A Fabric CA Server instance will be used to issue certificates to various nodes and clients within a particular organization. Fabric CA plays a key role when combining multiple ID's, as it is responsible for creating the basic X.509 certificates and their corresponding private keys, as well as setting the `hfa.ParentHash` and `hfa.ParentSignature` attributes if applicable.

Security Smart Contract The security smart contract is a custom-made smart contract that has two responsibilities.

- First, this smart contract is responsible for maintaining the "parent" X.509 certificates stored on the ledger, as described in Section 5.1. Clients that want to combine

[3]More detailed version available at https://github.com/daangordijn/Fabric-Access-Control/blob/master/images

two identities, e.g., identity A and identity B, have to invoke this smart contract with identity A. The smart contract will then calculate the SHA-256 hash of the provided certificate, store it in the hashmap on the ledger, and return the hash to the client. Now, the client can calculate the signature and set the required attributes.

- Second, this smart contract can be invoked by other smart contracts that live on the blockchain network to determine whether a client satisfies a particular access policy. Smart contracts can make use of the `ctx.stub.invokeChaincode()` method to invoke this security smart contract, provide the access policy that has to be validated, and will then be returned a boolean value indicating whether the client certificate satisfies the specified policy. The internal logic of this smart contract method is visualized on the right side in Figure 6.

Client Smart Contract(s) The client smart contracts are basic smart contracts that allow clients of the blockchain network to interact with the ledger. Examples of such smart contracts are the `asset-transfer` or `commercial-paper` chaincodes provided in the `fabric-samples` repository[4]. While previously, these smart contracts had to implement their business logic to validate whether a client has access to the requested resource, developers are now able to simply invoke the security smart contract using the `ctx.stub.invokeChaincode()` method of the Hyperledger Fabric SDK, and use the returned boolean to allow or deny the client from accessing the requested resource.

Client Application(s) The client applications are basic applications that allow clients of the blockchain network to more easily interact with smart contracts, instead of having to use the peer CLI. Examples of such client applications are the `asset-transfer` or `commercial-paper` applications provided in the `fabric-samples` repository[4]. To client applications, changes made to the proposed solution are not visible, except for the fact that some X.509 certificates containing valid `hfa.ParentHash` and `hfa.ParentSignature` attributes will now be granted access, while they would previously have been denied access from the network.

In summary, this section has presented a solution for combining multiple ID's, attributes, and policies in Hyperledger Fabric. Since this solution can be fully implemented using a single smart contract, the core components of the Hyperledger Fabric blockchain can remain unchanged. In the next section, a performance analysis will be presented, which analyses the increase in runtime due to the invocation and execution of the security smart contract.

6 Results

One of the most important considerations when proposing a new implementation is to minimize the latency and maximize the transaction throughput. To objectively analyze these performance indicators, two benchmarks of the implemented smart contract were performed with the help of the Hyperledger Caliper [33] blockchain benchmarking tool:

[4]Available at https://github.com/hyperledger/fabric-samples

- **Basic**: This benchmark analyzes the average latency and throughput when the entity that submits the transaction proposal can satisfy the access policy with its own X.509 attributes; and
- **Parent**: This benchmark analyzes the average latency and throughput when the entity that submits the transaction proposal can only satisfy the access policy with a parent certificate.

The exact configuration files that have been used to perform these two benchmarks can be found in the `caliper` directory of the public GitHub repository[2].

During this study, all benchmarks were performed on a virtual machine running Ubuntu 20.04 LTS, with a total RAM memory of 8GiB. The results that have been obtained are listed in Table 1 and visualized in Figure 7. All reports generated by Hyperledger Caliper can be found in the previously mentioned GitHub repository.

Checks (n)	Latency (Basic)	Latency (Parent)	Throughput (Basic)	Throughput (Parent)
1	0.04	0.05	93.3	85.9
10	0.04	0.05	98.8	90.7
50	0.04	0.05	103.0	94.1
100	0.04	0.05	100.9	89.9
500	0.06	0.06	82.5	75.4
1,000	0.07	0.07	68.3	66.8
5,000	0.17	0.19	30.0	26.1
10,000	0.31	0.37	18.0	15.1
50,000	1.51	1.66	3.9	3.3
100,000	3.12	3.40	1.8	1.5

Table 1: Average latency and throughput of the access control smart contract, measured using the Hyperledger Caliper benchmarking tool. Each row reports the measured latency and throughput associated with validating the submitted X.509 certificate on the defined access policy, which consists of n attribute checks.

As can be seen in the image, the average latency increases linearly with the number of attribute checks that have to be performed by the smart contract. On the contrary, the average throughput decreases exponentially with this same number of attribute checks. In addition, as can be seen in the image, the performance corresponding to satisfying the access policy with a parent X.509 certificate is slightly worse compared to satisfying this same access policy with its own attributes.

Finally, to objectively quantify these benchmark results, a base case was created and benchmarked using the same Hyperledger Caliper configuration. The smart contract method invoked during this base case benchmark immediately returned a Boolean value, without running any additional code. Hyperledger Caliper reported the average latency of this benchmark to be 0.04 seconds, and the average throughput to be 102.1 transactions per second. Comparing these values with the values listed in Table 1, it can be concluded that the increase in latency and decrease in throughput is very small. When keeping the number of attribute checks below 100, which is considered to be sufficient in most real-world applications, the decrease in performance can be disregarded.

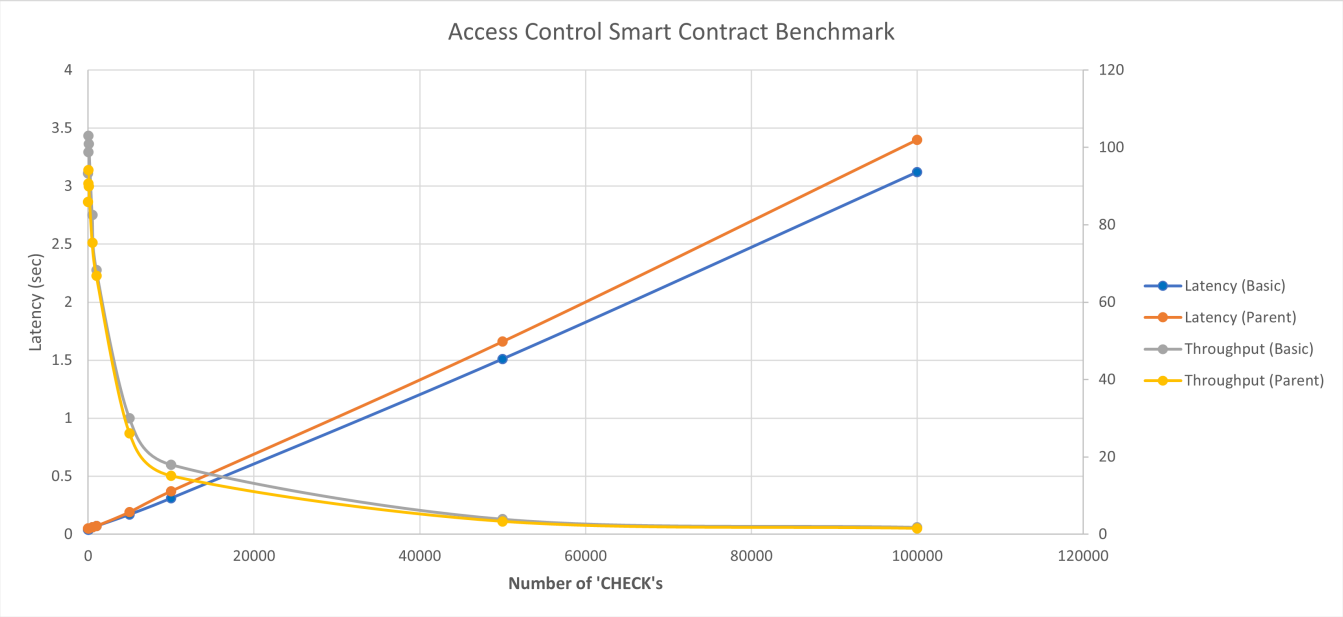

Figure 7: Average latency and throughput of the access control smart contract, measured using the Hyperledger Caliper benchmarking tool. The blue and grey lines respectively show the average latency and throughput that corresponds to the case where the submitting entity satisfies the access policy with its own attributes, while the orange and yellow lines show the case where the access policy had to be satisfied with the parent X.509 certificate, i.e., using the `hfa.ParentHash` and `hfa.ParentSignature` attributes.

7 Discussion

This section will discuss the process and technologies that were used to achieve the results and conclusions described in this paper. First, a literature study was conducted on existing literature in the field of secure access control within blockchains, and in particular Hyperledger Fabric. For this, several search queries have been defined, which are included in Appendix A, and executed using Google Scholar. Ultimately, it turned out that there was no existing literature that specifically related to the research topic of combining ID's, attributes, and policies within Hyperledger Fabric.

Next, a concept solution was defined which could be used to combine these IDs, attributes, and policies. Using Hyperledger Fabric's smart contract technology, these concepts were then implemented and rolled out on a local test network. Subsequently, a demo application and demo smart contract were implemented to analyze the behavior of these concepts. Finally, the implementation was benchmarked using Hyperledger Caliper, after which these results were analyzed and compared to a base case. Based on these results, it was concluded that for real-world scenarios, the performance impact caused by the implemented smart contract is minimal.

8 Conclusions and Future Work

Traditional permissionless blockchain technologies are not sufficient for enterprise-level applications, where privacy and trust are critical. Hyperledger Fabric solves this issue by being a permissioned blockchain technology and using concepts such as identities, channels, and private data collections to create this level of privacy and trust. One of the major problems of Hyperledger Fabric is that its current access control mechanism is not flexible enough for business scenarios. This study aimed to solve this issue by answering the research question "How can secure access control in Hyperledger Fabric be guaranteed by combining multiple ID's, attributes, and policies with the components that regulate access control?"

First, to combine multiple ID's within Hyperledger Fabric, a technique has been proposed that hashes and signs one certificate, referred to as the parent certificate, and adds this hash and signature as attributes to another certificate. In addition, this technique stores this full parent certificate in a hashmap on the distributed blockchain ledger. Upon receiving a transaction proposal, smart contracts on the blockchain can retrieve this parent certificate from the distributed ledger and verify ownership using the provided signature. Second, to combine multiple attributes, a technique has been proposed that created access policies using the policy check operators `EQUALS` and `INCLUDES`, which can be combined with the Boolean operators `AND`, `OR`, and `NOT`. Finally, to combine multiple policies, a technique has been proposed that maintains multiple policy definitions on the distributed ledger, which can dynamically be selected as the validating policy depending on the method invoked with the transaction proposal.

Finally, in terms of performance, it has been established that for real-world applications the performance impact is negligible. For access policies with less than 100 attributes to check, the increase in average latency is below 0.01 seconds compared to the base case of always allowing access. However, an increase in average latency of 0.1 seconds has been measured when comparing the case where the access policy is satisfied with a client's own attributes with the case where the access policy is satisfied with a client's parent certificate.

While this paper provides a fully working solution to solve the identified problem, some potential improvements have been identified that can be explored in future research:

- Although the benchmarks performed by Hyperledger Caliper indicate that the performance impact caused by the proposed implementation is minor, research could be done into ways of improving the algorithms used to validate access policies within the smart contract.
- Currently, the proposed implementation only allows users to set one certificate as its parent certificate using the `hfa.ParentHash` and `hfa.ParentSignature` attributes. Future research could be done to study whether multiple such parent certificates can be set, for example by allowing array-typed values for these two attributes.
- Although the proposed implementation allows users to define complex access policies by combining one or more `EQUALS` or `INCLUDES` operators using the `AND`, `OR`, and `NOT` operators, research could be done into ways of allowing users to define even richer access policies.
- Currently, clients must store private key data using file system wallets, which are considered insecure [34]. Future research could be done to allow users to store their private key data in Hardware Security Modules (HSM) to improve the security of this data.

9 Responsible Research

This section will reflect on the ethical aspects and the reproducibility of the study performed in this research paper.

9.1 Ethical Aspects

As described in Section 1, this study aims to present a way to provide secure access control by combining multiple ID's, attributes, and policies within Hyperledger Fabric blockchain networks. As a result, the most important ethical aspects relevant to this study are the security and privacy of the Hyperledger Fabric blockchain technology. While most traditional blockchain technologies do not provide ways to easily establish privacy and trust on the network, Hyperledger Fabric enables blockchain participants to identify each other through the use of identities. In addition, Hyperledger Fabric allows users to protect their data from other blockchain participants through the use of channels and private data collections. Because of these innovative techniques, Hyperledger Fabric provides all the basic features that traditional blockchains can offer, and builds upon these features to offer trust and privacy on the blockchain network.

An important point in the context of ethical responsibility is that this research aims to contribute to the security of Hyperledger Fabric by improving its access control mechanism. While access control must currently be performed by validating individual X.509 certificates within smart contracts, the solutions proposed in this paper allow smart contract developers to easily combine multiple ID's, attributes, and policies during their access control decisions. This allows smart contracts to make more fine-grained access control decisions, which in turn makes it easier to adhere to the principle of least privilege [35]. Therefore, the contributions of this paper only aim to improve the security and privacy of Hyperledger Fabric and can thus be considered ethically responsible.

9.2 Reproducibility

It is highly important that research is conducted in such a way that it is reproducible. By allowing other researchers to repeat the study presented in this paper, all contributions and results can objectively be validated. To guarantee the reproducibility of this particular study, several measures have been taken. First, Section 3 of this paper aims to provide an in-depth explanation of the methodology that was used to arrive at the final contributions. Second, the entire code-base that has been implemented as part of this study has been made publicly available on GitHub[2]. Finally, Section 6 presents the raw benchmark data that has been obtained, which can also be found in the `README` file inside the `caliper` directory of the aforementioned GitHub repository. Thus, since all research data is made publicly available, this study should be fully reproducible by other researchers.

A Search Queries

Combine the concepts with AND					
Combine the synonyms with OR	access control	Hyperledger Fabric	attributes	policies	ID's
	access control authentication	Hyperledger Fabric Hyperledger Fabric blockchain	attributes	policies	ID's IDs identities

Search Query 1	"access control" OR "authentication"
Search Query 2	"Hyperledger Fabric" OR "Hyperledger" OR "Fabric" OR "blockchain"
Search Query 3	"attributes"
Search Query 4	"policies"
Search Query 5	"ID's" OR "IDs" OR "identities"
Final Search Query	[search query 1] AND [search query 2] AND [search query 3] AND [search query 4] AND [search query 5]

Figure 8: Google Scholar search queries that have been used during the literature research, as described in [21]. First, five different search queries have been defined, which were then combined using the "AND" operator.

B Ledger Data Format

ObY1f2gGN3RFQ5wea2...	{ hash: "ObY1f2gGN3RFQ5wea2...", x509: "-----BEGIN CERTIFICATE-----\nMIIC3jCCAoSg..." }
2Kuf1grQpE3uAAJj8U...	{ hash: "2Kuf1grQpE3uAAJj8U...", x509: "-----BEGIN CERTIFICATE-----\nMIICzzCCAnQg..." }
d5A5OXY8mGerRuQfEo...	{ hash: "d5A5OXY8mGerRuQfEo...", x509: "-----BEGIN CERTIFICATE-----\nMIIDujCCA2Cg..." }
htcYg1ZvS4YlU9gNjB...	{ hash: "htcYg1ZvS4YlU9gNjB...", x509: "-----BEGIN CERTIFICATE-----\nMIIDujCCA2Cg..." }

"file1"	{ name: "file1.txt", cid: "Qmar2psoq8Z1FPJWn9uJ6hbt8aiWwi6dAQiT6hFBsHzf8C", policies: {...} }
"file2"	{ name: "file2.txt", cid: "QmX1j1LF9gNZ77iybf57CpCXaERCqD6pVqyXGFJDYoSCD6", policies: {...} }
"file4"	{ name: "file4.txt", cid: "QmWYyhxvrV7G1KH7rKGeukDkEzaZznarRa9gFo37DtLA9u", policies: {...} }
"file5"	{ name: "file5.txt", cid: "QmdqD4qq8ixFGutPd7UTgKTkgPXGcJdJX6ow8kWXSYPyux", policies: {...} }
"file6"	{ name: "file6.txt", cid: "QmWkRHp33CxXYbAM4uuRCLXjBfPNNb7sqTuRCv1t9PwpVw", policies: {...} }
"file8"	{ name: "file8.txt", cid: "QmXEdsQWJy97UCskfX46VzD4MJTzKgRbctL2Jiw5xpb88Z", policies: {...} }

Figure 9: Format of the data stored on the shared blockchain ledger. The first table represents the hashmap that stores X.509 certificates, as described in Section 5.1. The second table represents the storage of application data, in this case metadata of files stored on IPFS.

References

[1] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," *Bitcoin.org*, Oct. 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf.

[2] T. McGovern. "How Many Blockchains Are There In 2022?" (May 23, 2022), [Online]. Available: https://earthweb.com/how-many-blockchains-are-there/.

[3] J. Kelleher. "Why Do Bitcoins Have Value?" (May 19, 2022), [Online]. Available: https://www.investopedia.com/ask/answers/100314/why-do-bitcoins-have-value.asp.

[4] J. Frankenfield. "Smart Contracts." (May 19, 2022), [Online]. Available: https://www.investopedia.com/terms/s/smart-contracts.asp.

[5] D. Tapscott and A. Tapscott, *The Blockchain Revolution: How the Technology Behind Bitcoin is Changing Money, Business, and the World*. May 2016, ISBN: 978-0670069972.

[6] W. Cai, Z. Hong, Z. Wang, and C. Feng, "Decentralized Applications: The Blockchain-Empowered Software System," *IEEE Access*, Oct. 2018. [Online]. Available: https://www.researchgate.net/publication/327711685_Decentralized_Applications_The_Blockchain-Empowered_Software_System.

[7] M. Xu, X. Chen, and G. Kou, "A Systematic Review of Blockchain," *Financial Innovation*, Jul. 2019. [Online]. Available: https://jfin-swufe.springeropen.com/articles/10.1186/s40854-019-0147-z.

[8] L. Peng, W. Feng, Z. Yan, Y. Li, X. Zhou, and S. Shimizu, "Privacy Preservation in Permissionless Blockchain: A Survey," *Digital Communications and Networks*, May 2020. [Online]. Available: https://www.researchgate.net/publication/342455474_Privacy_preservation_in_permissionless_blockchain_A_survey.

[9] Hyperledger. "Hyperledger Fabric Documentation." (May 6, 2022), [Online]. Available: https://hyperledger-fabric.readthedocs.io/en/release-2.2/.

[10] National Institute of Standards and Technology. "X.509 Public Key Certificate." (Jun. 2, 2022), [Online]. Available: https://csrc.nist.gov/glossary/term/x_509_public_key_certificate.

[11] S. Ding, J. Cao, C. Li, K. Fan, and H. Li, "A Novel Attribute-Based Access Control Scheme Using Blockchain for IoT," *IEEE Access*, vol. 7, pp. 38 431–38 441, 2019. DOI: 10.1109/ACCESS.2019.2905846. [Online]. Available: https://ieeexplore.ieee.org/document/8668769.

[12] S. Rouhani and R. Deters, "Blockchain Based Access Control Systems: State of the Art and Challenges," in *IEEE/WIC/ACM International Conference on Web Intelligence*, ser. WI '19, Thessaloniki, Greece: Association for Computing Machinery, 2019, pp. 423–428, ISBN: 9781450369343. DOI: 10.1145/3350546.3352561. [Online]. Available: https://doi.org/10.1145/3350546.3352561.

[13] L. Song, M. Li, Z. Zhu, P. Yuan, and Y. He, "Attribute-Based Access Control Using Smart Contracts for the Internet of Things," *Procedia Computer Science*, vol. 174, pp. 231–242, 2020, 2019 International Conference on Identification, Information and Knowledge in the Internet of Things, ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2020.06.079. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050920315933.

[14] M. Yutaka, Y. Zhang, M. Sasabe, and S. Kasahara, "Using Ethereum Blockchain for Distributed Attribute-Based Access Control in the Internet of Things," in *2019 IEEE Global Communications Conference (GLOBECOM)*, 2019, pp. 1–6. DOI: 10.1109/GLOBECOM38437.2019.9014155.

[15] A. Iftekhar, X. Cui, Q. Tao, and C. Zheng, "Hyperledger Fabric Access Control System for Internet of Things Layer in Blockchain-Based Applications," *Entropy*, vol. 23, no. 8, 2021, ISSN: 1099-4300. DOI: 10.3390/e23081054. [Online]. Available: https://www.mdpi.com/1099-4300/23/8/1054.

[16] Z. Yang, D. Shao, L. Qu, and M. Zhang, "Internet of Things Access Control System Based on Hyperledger," *Journal of Physics: Conference Series*, vol. 1748, no. 4, p. 042 031, Jan. 2021. DOI: 10.1088/1742-6596/1748/4/042031. [Online]. Available: https://doi.org/10.1088/1742-6596/1748/4/042031.

[17] M. A. Islam and S. Madria, "A Permissioned Blockchain Based Access Control System for IOT," in *2019 IEEE International Conference on Blockchain (Blockchain)*, 2019, pp. 469–476. DOI: 10.1109/Blockchain.2019.00071.

[18] X. Zhao, S. Wang, Y. Zhang, and Y. Wang, "Attribute-Based Access Control Scheme for Data Sharing on Hyperledger Fabric," *Journal of Information Security and Applications*, vol. 67, p. 103 182, 2022, ISSN: 2214-2126. DOI: https://doi.org/10.1016/j.jisa.2022.103182. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2214212622000643.

[19] IPFS. "IPFS Documentation." (May 30, 2022), [Online]. Available: https://docs.ipfs.io/.

[20] H. D. Bandara, S. Chen, M. Staples, and Y. Sai, "Modeling Multi-Layer Access Control Policies of a Hyperledger-Fabric-Based Agriculture Supply Chain," in *2021 Third IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA)*, 2021, pp. 355–364. DOI: 10.1109/TPSISA52974.2021.00039.

[21] TU Delft Library. "Making A Search Plan." (May 30, 2022), [Online]. Available: https://tulib.tudelft.nl/searching-resources/making-a-search-plan.

[22] Hyperledger. "Using the Fabric Test Network." (May 30, 2022), [Online]. Available: https://hyperledger-fabric.readthedocs.io/en/release-2.2/test_network.html.

[23] ——, “What is Hyperledger Fabric?” (May 6, 2022), [Online]. Available: https://hyperledger-fabric.readthedocs.io/en/release-2.2/whatis.html.

[24] ——, “Hyperledger Fabric SDKs.” (May 30, 2022), [Online]. Available: https://hyperledger-fabric.readthedocs.io/en/release-2.2/fabric-sdks.html.

[25] Ethereum. “Smart Contract Languages.” (Jun. 8, 2022), [Online]. Available: https://ethereum.org/en/developers/docs/smart-contracts/languages/.

[26] Hyperledger. “Blockchain Network.” (Jun. 8, 2022), [Online]. Available: https://hyperledger-fabric.readthedocs.io/en/release-2.2/network/network.html.

[27] B. Lutkevich. “What Is Access Control?” (Jun. 8, 2022), [Online]. Available: https://www.techtarget.com/searchsecurity/definition/access-control0.

[28] M. Ed-Daibouni, A. Lebbat, S. Tallal, and H. Medromi, “Toward a New Extension of the Access Control Model ABAC for Cloud Computing,” in *Advances in Ubiquitous Networking*, E. Sabir, H. Medromi, and M. Sadik, Eds., Singapore: Springer Singapore, 2016, pp. 79–89, ISBN: 978-981-287-990-5.

[29] SSL.com. “What Is A Certificate Authority?” (Jun. 8, 2022), [Online]. Available: https://www.ssl.com/faqs/what-is-a-certificate-authority/.

[30] International Telecommunication Union. “Public-Key and Attribute Certificate Frameworks.” (Jun. 2, 2022), [Online]. Available: https://www.itu.int/rec/T-REC-X.509-201910-I/en.

[31] Hyperledger. “Fabric CA User’s Guide.” (Jun. 3, 2022), [Online]. Available: https://hyperledger-fabric-ca.readthedocs.io/en/release-1.4/users-guide.html.

[32] KeyFactor. “What is PKI and How Does it Work?” (Jun. 10, 2022), [Online]. Available: https://www.keyfactor.com/resources/what-is-pki/.

[33] Hyperledger. “Hyperledger Caliper.” (Jun. 17, 2022), [Online]. Available: https://hyperledger.github.io/caliper/v0.5.0/getting-started/.

[34] Solana. “Command Line Wallets.” (Jun. 18, 2022), [Online]. Available: https://docs.solana.com/wallet-guide/cli.

[35] M. Gegick and S. Barnum. “Least Privilege.” (Jun. 17, 2022), [Online]. Available: https://www.cisa.gov/uscert/bsi/articles/knowledge/principles/least-privilege.