RAPPORT DE STAGE

Soutenue à AMU — Aix-Marseille Université
le 11 septembre 2025 par

Anis NEMMICHE

How blockchain technologies can serve distributed access control

Discipline
Master Informatique

Spécialité
Fiabilité et Sécurité Informatique (FSI)

Laboratoire/Partenaires de recherche
- Laboratoire d'Informatique et des Systèmes
- Télécom SudParis

Encadrantes

Clara BERTOLISSI

Maryline LAURENT

Abstract

This report design and prototype a decentralized access-control framework for multi-cloud industrial collaboration. Built on Hyperledger Fabric and decentralized identifiers (DIDs), the system externalizes ABAC policies into modular smart contracts that evaluate Verifiable Presentations (VPs) derived from Verifiable Credentials (VCs) and issue single-use tokens. Sensitive files remain off-chain, the ledger stores hashes and audit logs and tokens control retrieval from organizational repositories. The prototype demonstrates on-chain policy evaluation, event logging and token-based cross-domain access. Results indicate that a permissioned blockchain combined with DIDs delivers auditable, fine-grained and interoperable access control across heterogeneous multi-cloud environments.

Keywords: Blockchain, Access control system, Cloud, Smart contracts, Decentralized Identifiers

Contents

Introduction

This research work was carried out within the *Laboratoire d'Informatique et Systèmes* (LIS) in Marseille, in collaboration with *Télécom SudParis*, under the supervision of Clara Bertolissi and Maryline Laurent. It was conducted as part of the national research project TrustInCloud, which aims to explore innovative mechanisms to strengthen trust, security and accountability in cloud environments. In this context, the internship focused on studying blockchain-based solutions for distributed access control in multi-cloud contexts.

The rapid adoption of cloud computing has significantly transformed the way organizations store, share and manage sensitive information. However, this transformation also introduces major challenges, such as data breaches, privilege escalation attacks and a lack of transparency around accountability. Traditional access control mechanisms, such as role-based access control (RBAC) and attribute-based access control (ABAC), are often centralized and rely on a trusted authority to enforce policies. While these models have proven effective in closed environments, they are less suited to open, distributed ecosystems such as the cloud, where multiple independent stakeholders interact and access conditions can change dynamically. Blockchain technology, with its decentralized and tamper-proof ledger, offers promising possibilities for overcoming these limitations. By embedding access control rules directly into smart contracts, it becomes possible to decentralize decision-making, automate policy enforcement and ensure the integrity and traceability of all access attempts. This creates new opportunities to design secure frameworks in which organizations can cooperate while retaining control over their data and ensuring verifiable accountability for all actions.

The central problem investigated in this work is therefore how blockchain technologies can serve distributed access control, particularly in multi-cloud environments. The goal is to determine how smart contracts and decentralized identity mechanisms can be combined to provide fine-grained, auditable and context-aware access management, while preserving confidentiality and interoperability between organizations.

1 Preliminaries

This section introduces the fundamental concepts necessary for understanding the approach proposed in this work. We present the principles of blockchain technology and the main traditional access control mechanisms, as well as the evolution towards multi-cloud environments, which highlights the need for new distributed approaches.

1.1 Cloud and Multi-Cloud Environments

Cloud computing has fundamentally changed how organizations manage their IT resources. It provides access to a variety of services, including infrastructure, storage and applications. The benefits and flexibility of cloud computing have motivated organizations to migrate their on-premises resources to a cloud-based solution. Cloud environments allow organizations to optimize operational costs, improve service availability and accelerate deployment cycles. Additionally, resources are managed dynamically and on demand rather than physically, offering better scalability.

However, while adopting cloud services may open up new opportunities for organizations, it has also created new challenges, particularly with regard to security, privacy and access control. Since resources are hosted externally, organizations must trust third-party providers to protect sensitive data. Dependence on external providers also raises the significant issue of vendor lock-in. To mitigate these risks and gain greater control over their digital assets, many organizations have adopted multi-cloud architectures.

In multi-cloud environment, services and data are distributed across multiple independent cloud providers. This approach enhances system resilience and optimizes performance and costs by leveraging the strengths of different providers, while reducing dependency on a single provider. However, transitioning from a single-domain cloud to a multi-domain environment introduces an additional layer of complexity in governance and security management. While the access control policy is centralized in a single-domain cloud, each cloud service provider has its own security policy in a multi-domain cloud, which complicates interaction between them.

Therefore, maintaining consistent access control and coherent identity management through multiple entities in multi-domain cloud environments, becomes extremely difficult. New approaches are needed to reinforce distributed networks and guarantee policy interoperability. Blockchain technologies, with their inherent properties of decentralization, transparency and tamper-proof, offer a promising prospects for addressing these challenges.

1.2 Blockchain

Blockchain is a decentralized and tamper-proof ledger technology that is designed to securely record transactions across a network of distributed nodes. Transactions are recorded in blocks, each of which is cryptographically linked to the previous one via its hash to form an immutable chain. Since the hash of each block is derived from the previous one, altering a single block necessitates modifying all subsequent ones, rendering the ledger resistant to unauthorized alterations. These characteristics ensure data integrity, transparency and traceability. Blockchains are relevant in environments where trust is distributed and no central authority is responsible for validating operations. Consequently, they are being adopted increasingly in domains such as finance, education, supply chain, healthcare and cloud computing where auditability, resilience and integrity are required.

There exist three main types of blockchain infrastructures, each with different characteristics :

Public blockchains, such as Bitcoin and Ethereum, are fully decentralized and open to any participant. Anyone can join the network, submit transactions and participate in consensus (e.g., Proof of Work). These systems are highly transparent and distributed, but suffer from limited transactions speed and high energy consumption.

Private blockchain restrict access to authorized participants only. The entity operating the blockchain manages access rights, transaction validation, and ledger updates. While they offer improved performance and privacy, they centralize control within a single trusted authority.

Permissioned blockchain combine the features of both private and public blockchains. Participation is restricted to authorized entities, which must be validated before they can join the network. Once admitted, participants can perform actions according to their assigned permissions, each user's permissions are limited to the roles and right granted by the network governance. This system offers fine-grained access control, high performance and enterprise integration.

The choice between public, private and permissioned blockchain architectures depends on several criteria. Figure 1.1 presents a decision diagram by Wüst and Gervais Wüst et al. 2018, that helps determine the most appropriate type of blockchain based on system requirements.

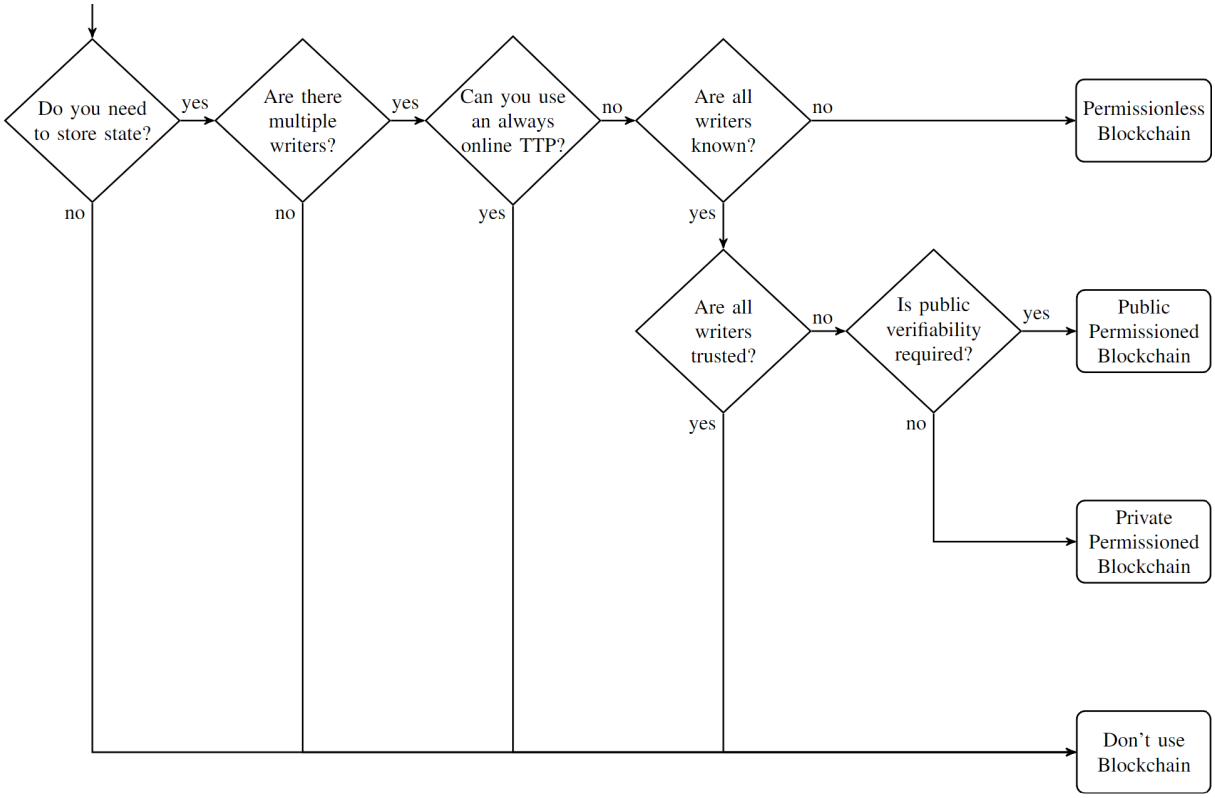

Figure 1.1 – Decision diagram for choosing a blockchain type based on system requirements, Wüst et al. 2018

Consensus mechanisms are a fundamental component of blockchain systems, ensuring that all nodes agree on the current state of the ledger. Public blockchain such as Bitcoin rely on **Proof of Work (PoW)**, where participants, called miners, must solve complex mathematical problems requiring significant computing power in order to validate transactions. This mechanism is robust and secure but comes at the cost of extremely high energy consumption and limited throughput. Other systems such as Ethereum have adopted **Proof of Stake (PoS)**, in which validators are selected based on the amount of cryptocurrency they "stake" as collateral. PoS significantly reduces energy requirements but introduces new challenges related to governance and economic fairness, since influence in the system is directly tied to financial commitment. In contrast, permissioned blockchains like Hyperledger Fabric adopt a different model based on endorsement and ordering. In Fabric, specific peers, called endorsers, validate transactions against predefined policies, while ordering nodes sequence the endorsed transactions into blocks. This separation of roles allows Fabric to achieve higher throughput, lower latency and fine-grained governance making it particularly suitable for enterprise contexts, where efficiency, accountability and controlled participation are required.

Permissioned blockchains have demonstrated significant advantages in terms of privacy, performance and enterprise integration. By supporting optimized consensus algorithms, fine-grained

access control and reduced transaction latency, Fabric provides a robust infrastructure tailored to organizational needs. This flexibility is largely driven by Fabric's integration of smart contracts, also know as "chaincode" within Fabric framework. These allow business logic and access policies to be implemented directly on the blockchain.

1.2.1 Smart contracts

Smart contracts are digital contracts stored in a blockchain that are automatically executed when the terms of the contract are met, without the need for intermediaries or centralized authorities. These contracts follow a simple logic, based on "if...then..." written in the blockchain code. When the predefined conditions are met, the contracts automatically trigger the associated actions.

In the context of access control, a smart contract can, for example, verify whether a user satisfies the required attributes or permissions before granting access to a resource. Beyond simple logic execution, smart contracts also provide accountability. Every decision, whether granting or denying access, is recorded in the immutable ledger, creating verifiable logs that enhance auditability and ensure that all stakeholders share a transparent and consistent source of truth for access control enforcement.

1.2.2 Decentralized Identifiers (DIDs)

DIDs are a new type of identifier defined by the W3C standard that enables verifiable, self-sovereign identities without relying on a central authority or identity provider. Unlike traditional identity management models, where identifiers are issued and validated by trusted intermediaries, DIDs are created, owned and managed directly by the entity they represent, whether it is a person, an organization or a device. A DID is associated with a DID Document, which contains metadata such as public keys, service endpoints and authentication methods. This document can be accessed through a resolver according to the DID method in use, allowing verifiers to validate ownership and authenticity.

From a functional perspective, DIDs are designed to support both identification and authentication. For example, when a holder presents a credential, the verifier can confirm ownership through a challenge response mechanism based on the cryptographic keys linked to the DID. This process ensures that only the legitimate controller of the identifier can prove its validity. Several DID methods coexist today, reflecting different design choices and infrastructures. Publicly registered DIDs, such as *did:ethr* on Ethereum, leverage blockchain transactions to publish and resolve DID documents. Others, such as *did:web*, rely on existing web infrastructure and DNS resolution, while *did:key* operate in a purely peer-to-peer context without requiring a global registry. Each method involves specific trade-offs in term of privacy and security.

The privacy dimension is essential to the adoption of DIDs. As highlighted in Naghmouchi et al. 2025, privacy-preserving features as pseudonymity, selective disclosure and unlinkability are essential to prevent decentralized identity systems from becoming surveillance tools. Mechanism such as zero-knowledge proofs enhance for the ability of holders to use multiple identifiers in different contexts without them being correlated. This is part of the privacy-by-design paradigm, which puts the user in control of how, when and with whom identity attributes are shared.

In practice, DIDs provide the foundation fo verifiable credentials and decentralized access control. They enable a secure, interoperable and user-centric identity layer that is increasingly integrated into emerging standards, such as the EU digital identity framework. In the context of blockchain-based access control, DIDs serve as trusted anchors for participants, ensuring that only authenticated and authorized entities can request or exchange documents across distributed and multi-cloud infrastructure.

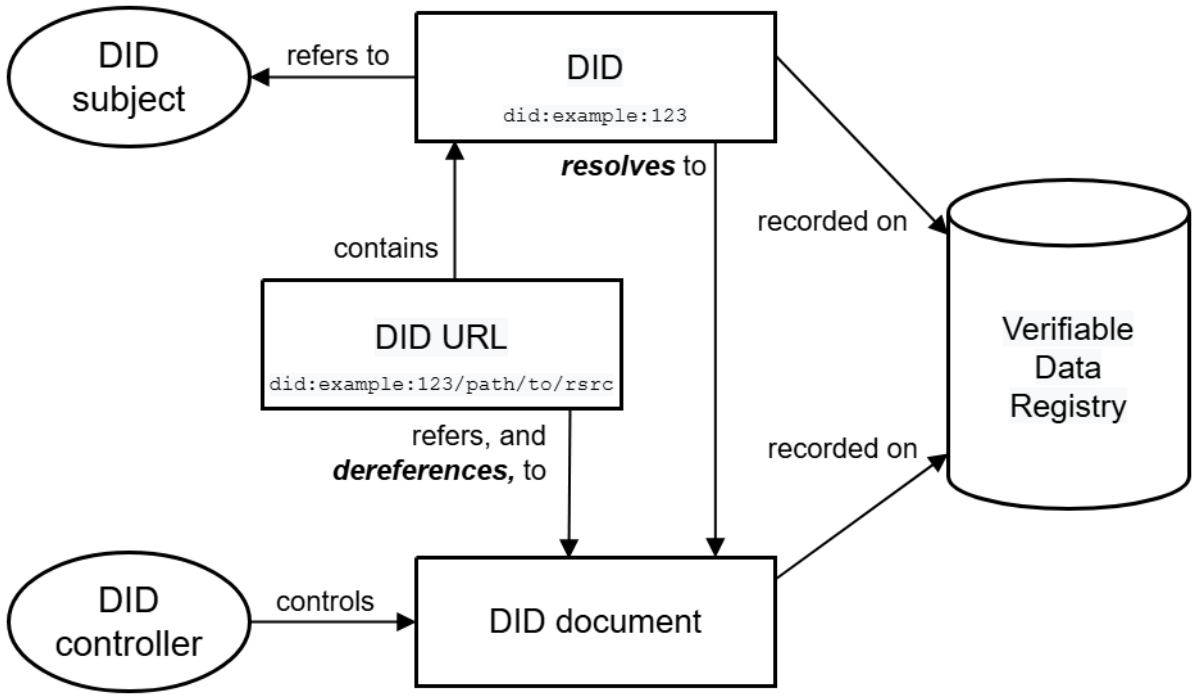

Figure 1.2 – Overview of DID architecture and the relationship of the basic components (adapted from W3C DID v1.1 specification)

The overall functioning of a DIDs is illustrated in Figure 1.2. In this model, a **DID subject** is the entity being identified. The **DID** itself, *did:example:123*, is recorded on a **Verifiable Data Registry**, typically a blockchain or another distributed system. Each DID resolves to a **DID document**, which contains the cryptographic material and service endpoints needed to authenticate and interact with the subject. A **DID controller** manages this DID document and can update or revoke its content. Finally, a **DID URL** can be used to refer to a specific element of the DID document, such as a key or a service endpoint, and resolving it provides the verifier with the necessary information to validate proofs or establish secure communication.

1.3 Access control

Access control systems define rules for determining who can access which resources under what conditions. In traditional IT systems, access control is typically enforced using centralized mechanisms. The main models are described below.

Discretionary Access Control (DAC) allows data owners to define access policies according to their own discretion. Although this model is simple and flexible, it requires a central authority. In DAC systems, each object has an owner, who can grant or revoke access permissions to other users. However, DAC systems does not allow global policies to be applied and are vulnerable to unintentional propagation of privileges.

Mandatory Access Control (MAC) applies access policies based on predefined classifications and security labels. In MAC systems, a central authority determines access decisions by defining access policies to the data. Each user and resource is assigned a security level (e.g., Public, Confidential, Secret or Top Secret), and access is only granted if the user's authorization level meets or exceeds the required classification. MAC is commonly used in military, government or critical infrastructure systems where the control of information must be strict.

Role-Based Access Control (RBAC) manages access permissions based on the roles assigned to users within an organization. RBAC associates permissions with roles (e.g., Administrator), and users inherit the access rights of the roles they are assigned to. This approach makes access control management more efficient and reduces human error.

Attribute-Based Access Control (ABAC) determines access decisions based on the evaluation of attributes associated with users, resources, actions, and the environment. These attributes can represent a diverse identity information or contextual data such as roles, groups, locations and the time of the request and more. In ABAC systems, policies are defined by combinations of attributes. This allows a high degree of flexibility and precision in policy implementation.

Several recent studies also highlight the potential of blockchain to strengthen these traditional models by decentralizing policy enforcement and ensuring tamper-resistant auditability Wijesekara 2024. Figure 1.3 illustrates a representative blockchain-based architecture with an access control management layer for cloud environments.

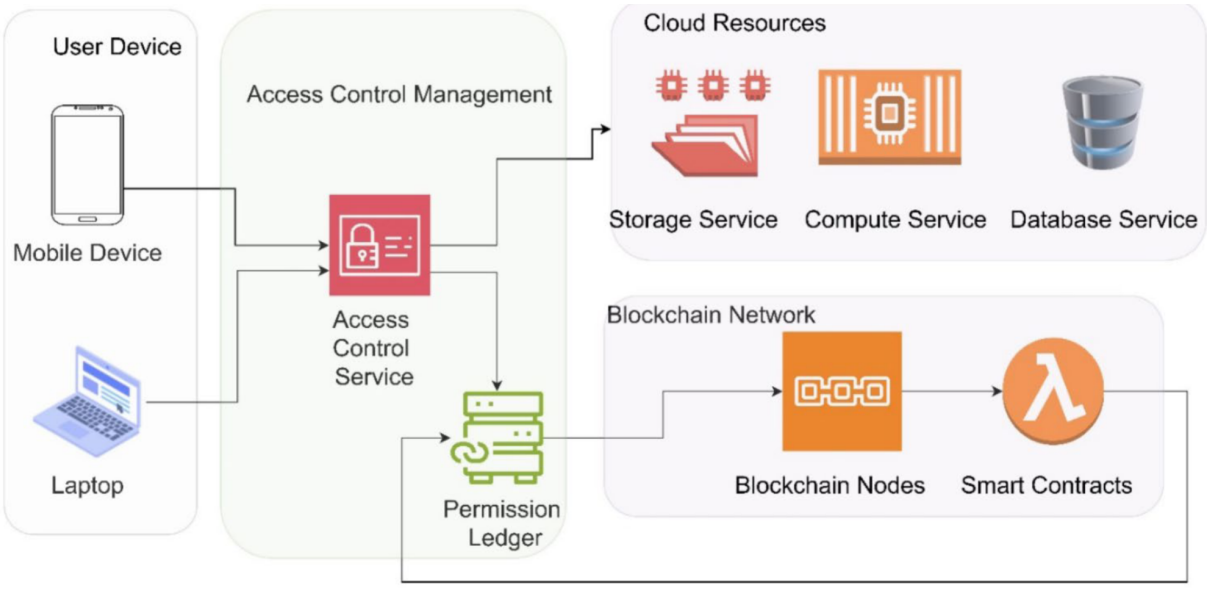

Figure 1.3 – The block diagram of blockchain-based access control, Punia et al. 2024

1.4 Internship Objective

The objective of this internship is to design and prototype a blockchain-based Attribute-Based Access Control framework for multi-cloud collaboration. The framework must :

— externalize policy evaluation in smart contracts
— bind access requests to DID and Verifiable Credential (VC)
— and provide auditability across organizational boundaries.

2 State of Art

Today, organizations are increasingly required to collaborate with each other, relying more and more on cloud-based platforms for sharing data. While these platforms offer many advantages in terms of cost and scalability, they also present challenges with regard to confidentiality, data security and governance. In particular, companies need to implement a consistent access control model across multiple untrusted entities. Although traditional access control models such as RBAC and ABAC are effective in a single-domain context, they are significantly limited in a cross-domain context. This is mainly because they rely on a centralized point for policy decisions and trusted authorities, which is not viable when multiple independent organizations are involved. Furthermore, these models lack transparency and auditability. The absence of a trustworthy access control layer across organization leads to fragmented security policies, inconsistent enforcement, and an increased risk of data breaches or abuses. To address these challenges, there is a growing interest in decentralized access control that can operate across multiple domains without relying on centralized trust anchors. Blockchain technology is increasingly recognized as a foundational component in enabling this transition, providing a decentralized, tamper-resistant and transparent infrastructure for managing policies, identities and access logs. Several studies have examined the integration of blockchain into access control architectures to overcome these limitations by enabling secure, traceable, fine-grained access across multiple cloud domains. These approaches leverage blockchain in combination with cryptographic primitives, such as Attribute-Based Encryption (ABE), to enforce decentralized access control across untrusted domains. However, they differ in their architecture and cryptographic foundations.

Li et al. 2022 propose the **Cloud-Blockchain Fusion Framework (CBFF)**, which ensures data accountability in multi-cloud environments. The CBFF framework introduces a unified **Naming and Addressing Mechanism (NAM)** to globally identify cloud resources, as well as an **Operation Tracing Mechanism (OTM)** to log data operations in blockchain using Hyperledger Fabric. The framework uses symmetric encryption to secure data before uploading it to the cloud. The encrypted data is stored off-chain, while metadata such as the file hash and operational logs are recorded on-chain using smart contracts. This enables authorized users to easily locate a resource through their global identifier, verify its integrity using the stored hash, on the blockchain, and decrypt it locally with their symmetric key, ensuring both data confidentiality and operational traceability.

Park et al. 2024 propose a **Cross-Domain Bilateral Access Control model** for data trading in cloud systems, aiming to establish trust between data senders and receivers. This access control scheme is based on **Identity-Based Matchmaking Encryption (IBME)**, which enables both parties to specify access conditions during the encryption process. In this model, the sender defines a set of attributes that the receiver must possess to decrypt the data, and the receiver specifies the properties required of the sender. Encryption and decryption are only successful if the conditions of both parties are mutually satisfied, ensuring bilateral access. Data is stored encrypted in the cloud, and all identity verifications and policy checks are managed via smart contracts deployed on a blockchain.

Liu et al. 2023 propose an access control model designed to enable secure, and fine-grained data sharing across organizations using a consortium blockchain. This model is based on unified attribute system across domains. Access decisions are enforced through smart contracts on Hyperledger Fabric, which check whether the user's attributes satisfy the policy defined by the data owner. The encrypted data is stored off-chain using **IPFS (InterPlanetary File System)**, while the metadata and access control logic remain on-chain to ensure transparency and traceability. This approach facilitates interoperability between organizations while preserving data confidentiality and auditability.

Shih et al. 2022 propose distributed access control model for the Industrial Internet of Things, built on Hyperledger Fabric and relying on smart contracts to manage access policies. The architecture introduces three distinct contracts: a **Policy Contract (PC)** to define and validate ABAC-based rules, a **Device Contract (DC)** to register resources generated by industrial equipment and produce one-time URLs, and an **Access Contract (AC)** responsible for evaluating user requests. By combining blockchain and ABAC, this model provides dynamic, decentralized, and auditable access control tailored to distributed industrial environments. The use of temporary URLs enhances security by preventing data reuse or exposure, while the recording of access decisions on the blockchain guarantees immutability and traceability.

While the reviewed approaches highlight the potential of blockchain to strengthen access control in distributed and multi-cloud environments, they also exhibit certain limitations. Most frameworks rely on static or pre-defined policies that are difficult to adapt in highly dynamic environments, where participants and requirements evolve continuously. They all require a prior agreement between participating organizations to define common policies, attributes and trusted entities, which introduces rigidity and limits autonomy. Current approaches also lack support for decentralized trust governance and scalable cross-domain enforcement, making them less suitable for open and evolving ecosystems. In many cases, access rules are still fixed by a central authority or consortium, which reduces flexibility and may not reflect the autonomy of each organization involved. To address these challenges, we propose a blockchain-based access control model for multi-cloud environments designed to support flexible policies, decentralized identity and autitable enforcement across organizations.

3 Proposed Model

Building on the limitations identified in the state of the art, this work proposes a blockchain-based access control model designed to support dynamic, decentralized and verifiable policy management across multiple cloud domains. Unlike existing approaches, which often require prior agreement between organizations to establish common policies and trusted authorities, the proposed model allows each consortium participant to define and enforce its own access control rules. These policies are managed through smart contracts deployed on a permissioned blockchain.

The model is organized around four key elements:

— **A permissioned blockchain**, built with Hyperledger Fabric
— **Smart contracts**, implementing Attribute-Based Access Control (ABAC)
— **DID** and **VC** for identity management
— **Off-chain storage**, for sensitive data secured by token-based access

In a interconnected industrial environment, several stakeholders must collaborate and exchange sensitive information. The **manufacturer** acts as the central producer of technical plans and specifications that must be shared with trusted partners. These documents are critical assets that often contain intellectual property and therefore requires strict access control. The **subcontractor**, who relies on these documents to execute specific production tasks, needs secure access to only the information related to its assigned contracts. Finally, the **certifiers** are responsible for verifying compliance, auditing processes and ensuring that quality and safety standards are met. Their role requires access to selected reports but never to the full set of proprietary design documents.

The model proposed in this work addresses these challenges through a decentralized architecture in which access rules are embedded in smart contracts executed in a transparent and tamper-resistant manner. Identities are managed using a DID system, while access rights are transmitted through VCs issued by trusted authorities. This combination ensures that manufacturers retain control over sensitive assets, subcontractors receive only the information they are entitled to, and certifiers gain verifiable evidence for compliance without exposing unnecessary data.

3.1 Use case

We consider three actors, each in a distinct domain: a **manufacturer**, a **subcontractor** and a **certifier**. The manufacturer must keep full control of technical plans while logging every access for traceability. The subcontractor must see only the information needed for its contractual tasks. The certifier must verify compliance. In our scenario, the manufacturer stores plans in its own repository and hashes on the permissioned ledger. It issues a ContractID and a VC to the subcontractor and presents these to a smart contract that enforces ABAC policy. If the attributes match, the smart contract delivers a time-bounded token. The subcontractor uses that token to fetch the plans from the manufacturer's repository. The model also supports reverse flows : the manufacturer can request production reports from the subcontractor's domain under the same access control and token pattern, and the certifier can read audit reports. The approach scales to multiple manufacturers, subcontractors and certifiers without changing the trust model.

For the **manufacturer**, the goal is to guarantee full control over sensitive technical plans and intellectual property. The model ensures that documents can be shared securely with trusted partners, while every access attempt, successful or not, is immutably recorded on the blockchain. This provides manufacturers with strong guarantees of traceability and protection against unauthorized disclosure. The manufacturer's policy accepts a request only if it comes from a requester whose role and organization are designated by the manufacturer for the specific Contract referenced in the request.

For the **subcontractor**, the objective is to enable access to only the information to its contractual tasks. By relying on attribute-based access control policies encoded in smart contracts, the model ensures that subcontractors can retrieve precisely the data to which they are entitled, without exposing

unrelated resources. This fine-grained control improves operational efficiency while reducing security risks. The subcontractor's policy accepts a request only if it comes from a requester whose role and organization are designated by the subcontractor for the specific ContractID referenced in the request and only for documents within the scope of that contract.

For the **certifier**, the model aims to provide transparent and verifiable access to compliance-related documents. Certifiers can authenticate themselves using DIDs and present verifiable credentials proving their role and authority. This allows them to access audit reports or certifications without exposing proprietary design documents. In addition, all their actions are logged on-chain, reinforcing accountability and trust in the certification process.

3.2 Objectives of the Model

The primary objectives of the proposed model are to establish a secure and trustworthy framework for document exchange between multiple organizations within a shared industrial ecosystem. In such ecosystems, several independent organizations interact, each with distinct requirements regarding confidentiality, integrity and accountability. The model therefore seeks to meet these requirements by combining blockchain technology, smart contracts and decentralized identity management.

To define the scope of the model, we assume that all consortium participants behave honestly and that malicious actors, external security breaches or infrastructure failures are outside the scope of this work. The focus is placed on access control mechanisms rather than on resilience against Byzantine faults or attacks. The model targets on two key security mechanisms :

— **Authentication**, by ensuring that each request is bound to a verifiable decentralized identity and supported by valid credentials.
— **Access control**, by enforcing fine-grained, policy-based rules embedded in smart contracts and applied consistently across organizations.

The choice of a **permissioned blockchain** as the blockchain platform follows from these requirements. It restricts participation to authorized organizations and supports governance models aligned with industrial consortium. Its modular design enables flexible consensus mechanisms, built-in privacy for data exchange and native support for smart contracts. These characteristics make it particularly well suited for secure collaboration across multiple organizations in a multi-cloud context.

In summary, the proposed model has three objectives : to protect manufacturers' intellectual property, to provide controlled and relevant access to subcontractors and to enable transparent and controllable verification. The following section presents the architecture of the model, detailing its main components and how they interact to achieve these objectives.

3.3 System Architecture

The proposed architecture combines blockchain-based access control mechanisms with decentralized identity and secure off-chain storage, designed to operate in a multi-cloud environment. It is designed to support industrial ecosystems where manufacturers, subcontractors and certifiers must collaborate.

3.3.1 System Components

The architecture relies on a set of core components that together ensure decentralized trust, secure identity management and controlled access to resources. Figure 3.1 provides a schematic overview of the model.

— **Permissioned blockchain network (Hyperledger Fabric)** : acts as the trust layer among all participating entities. It maintains an immutable ledger for storing document metadata and activity logs.
— **Smart contracts (chaincode)** : implement policy enforcement, access validation and token generation. Each request from the user is submitted with a **Verifiable Presentation (VP)** and a **ContractID** as input. Then, the smart contract validates the VP, checks the access control policy and records the decision on the blockchain to ensure full traceability. If the request is approved, the smart contract generates a **single-use**, **time-limited access token** based on the previously provided ContractID, which the authorized user can present to the off-chain storage system to retrieve the corresponding documents.
— **Decentralized identity management** : based on DIDs and VCs. Trusted entities, such as the manufacturer, issue credentials that certify the roles and attributes of participants (e.g., subcontractor). During access requests, these credentials are transformed into VP, which are cryptographically verifiable and prove ownership of one or more VC. Unlike the VC, VP allow the holder to select which information to disclose, thereby enhancing privacy and enabling smart contracts to enforce attribute-based policies in a decentralized way.
— **Off-chain repositories** : sensitive data are stored in private, public or hybrid cloud infrastructures controlled by each organization. The blockchain only records the metadata and hashes of these documents, allowing their integrity to be verified without exposing their content. Access is mediated by the token issued by the smart contracts. However, access is not limited to the use of this token alone, but also by another authentication on the storage system. The token is intended to ensure traceability and provide an additional layer of security.
— **Web portal** : a web interface for consumer to verify the authenticity of a purchased product by scanning a QR code or entering a serial number. The portal performs read-only queries to the blockchain to fetch the on-chain proofs (e.g., certificate, batch, timestamps) and displays the verification result.

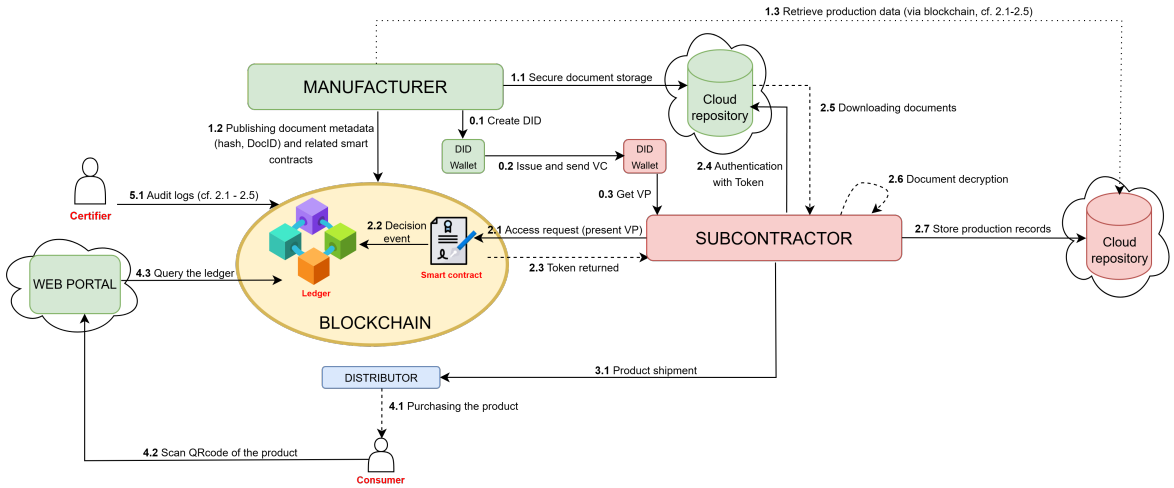

Figure 3.1 – System architecture

3.3.2 Actors

Manufacturer : the entity responsible for generating and sharing assets. It uploads technical documents to its private repository, registers metadata on the blockchain and issues VCs to trusted partners.

Subcontractor : the actor that relies on the manufacturer's documents to carry out assigned operations. It submits requests for documents linked to a specific contract, presenting its VP and ContractID. Upon validation, it receives a token granting access to only the data relating to its assigned tasks.

Certifier : the auditing authority. It verifies compliance by requesting access to selected reports. Its requests and activities are immutably logged on-chain, ensuring accountability and transparency.

Distributor : external sales actors. They don't interact with the blockchain, they simply transport and sell products.

Consumer : external to the consortium. They can uses the public verification portal to check product authenticity.

This section has outlined the architecture of the model. In the next section, we will present the interactions between components and actors, illustrated through sequence diagrams of the main workflows.

3.4 Overview of How It Works

This section illustrates the operational workflow of the proposed model through three main phases, each represented by a sequence diagram. Together, they describe the initialization of the system, the registration and discovery of documents and the secure retrieval of resources through token-based access.

System Initialization

In the initialization phase (Figure 3.2), the manufacturer (or another trusted entity) deploys the smart contracts on the blockchain network and defines the access control policies to be enforced. At the same time, DIDs are assigned to participating entities and VCs are issued to certify their roles and attributes. These credentials form the basis for all subsequent access requests.

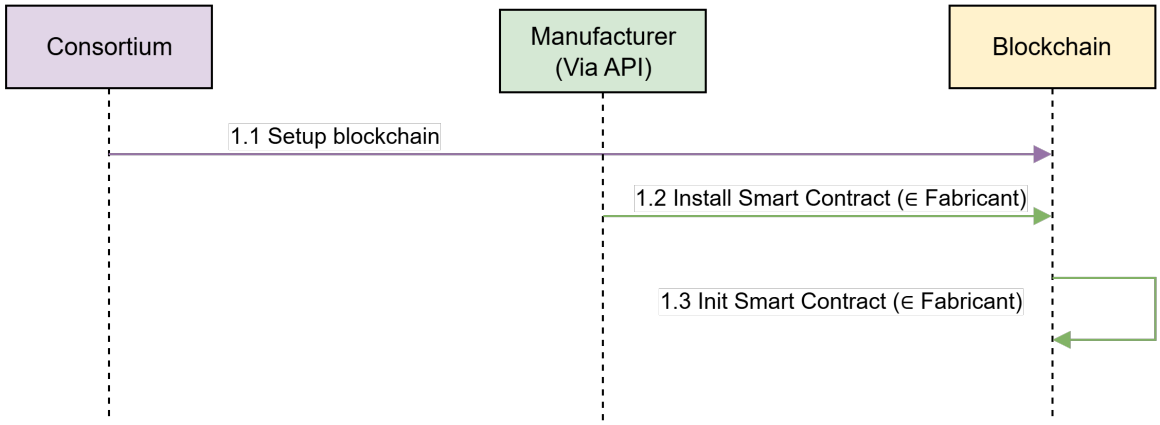

Figure 3.2 – System Initialization

Document Registration and Query

When a new document is produced, the responsible organization submits its metadata, consisting of the Hash-DocID pair and the associated ContractID, to the previously deployed smart contract and stores the document in its own database (off-chain). Access requests from different partners wishing to retrieve documents are submitted to smart contracts, providing the ContractID and a VP as input parameters. The smart contract verifies the authenticity and validity of the VP, checks the access control policy associated with the specified contract, and if the request is approved, returns the corresponding list of document metadata stored on the blockchain as well as an access token to retrieve files from off-chain storage (Figure 3.3).

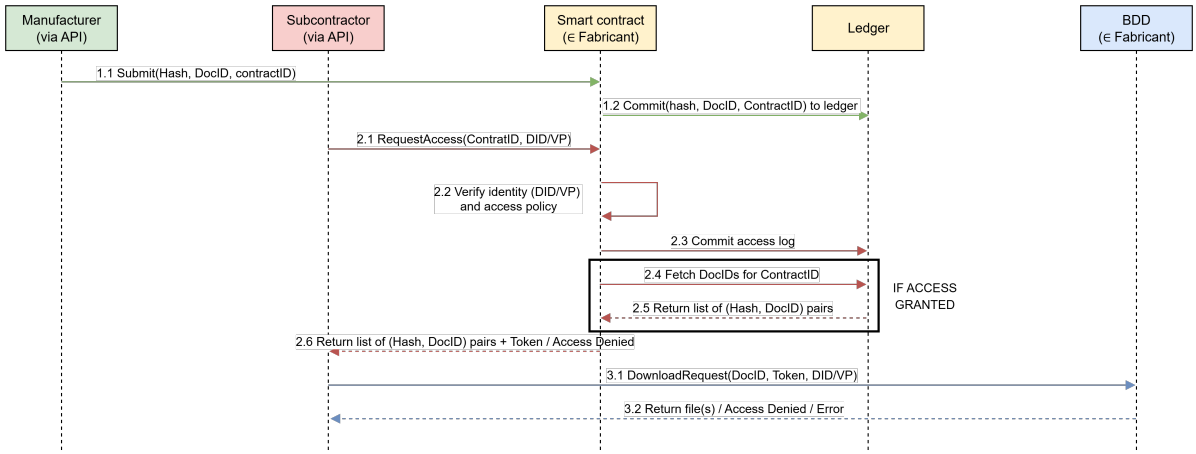

Figure 3.3 – Document Access Workflow

Secure Retrieval via Token-Based Access

As illustrated in Figure 3.3, once the access token has been issued, the authorized partner initiates the retrieval process by authenticating to the off-chain storage system with a VP. Alongside the VP, the partner provides the access token previously generated by the smart contract. This mechanism acts as a form of two-factor authentication : the VP confirms the partner's identity while the token validates the specific access request and its context. The storage system verifies both elements, ensuring the token has not expired and has not been previously used. If all checks succeed, the system grants access to the corresponding encrypted files. The integrity of each file can then be verified by comparing its locally computed hash with the hash value stored on the blockchain, guaranteeing that the content has not been altered.

3.5 Access Control model

The proposed model relies on Attribute-Based Access Control (ABAC) policies encoded within smart contracts. ABAC provides flexibility by evaluating multiple attributes from the subject, the resource and the environment.

3.5.1 Policy Specification

Access decisions are based on the evaluation of attributes belonging to users, resources and the context. Each domain defines and manages its own attributes, which are then used to evaluate access requests. In our setting, domain are the manufacturer, the subcontractor and the certifier, each actor has their own domain. Formally, attributes are defined as follows :

User attributes :

$$Attr(u) = \{Role, Organization\}$$

These attributes are derived from VCs. They verify the role of the requester, the organization it belongs to and the validity of its credentials. Here, Role ∈ {manufacturer, subcontractor, certifier}, and organization denotes the requester's domain identifier, with Organization ∈ {Org1, Org2, …, OrgN}.

Resource attributes :

$$Attr(r) = \{ContractID, DocID\}$$

Each document is linked to a specific identifier. The ContractID defines the scope of access and ensures that the requester can only access the documents for its assigned contracts. Here, ContractID denotes the contractual scope binding the resource to a specific contract and DocID denotes the unique identifier of the document, with ContractID ∈ {Contract1, Contract2, …, ContractN} and DocID ∈ {Doc1, Doc2, …, DocN}.

The general access control policy can be specified using a decision function that evaluates whether the attributes of the requester satisfy at least one of the policies defined by the resource owner. Consider $\boldsymbol{P} = \{p_1, p_2, \ldots, p_n\}$, the global set of access control policies across the consortium domains. Each element $p_i \in P$ encodes specific conditions over user attributes and resource attributes that must be satisfied for access to be granted. Each policies p_i are recorded on-chain inside a smart contract (Policy Contract) and evaluated by the Policy Decision Contract via the checkAccess $\big(\text{Attr}(u), \text{Attr}(r), p_i\big)$ function.

$$\text{Decision}(u, r) = \begin{cases} \text{Permit} & \text{if } \exists p_i \in P : \text{checkAccess}\big(\text{Attr}(u), \text{Attr}(r), p_i\big) = \text{ true} \\ \text{Deny} & \text{otherwise} \end{cases} \tag{1}$$

where

$$\text{checkAccess}\big(\text{Attr}(u), \text{Attr}(r), p_i\big) = \big(\text{Attr}(u) \in p_i\big) \wedge \big(\text{Attr}(r) \in p_i\big) \tag{2}$$

In our model, p_i can be instantiated as follows :

$$p_i = (\text{role} = \text{subcontractor} \wedge \text{organization} = \text{Org2}) \tag{3}$$

$$\wedge\ (\text{ContractID matches the credential}) \tag{4}$$

Thus, a request is granted only if the requester possesses valid credentials and the provided ContractID corresponds to its assigned contract. Otherwise, the request is denied.

3.5.2 Smart Contract Structure

The contract layer is structured into three sub-components, as suggested by various articles like Shih et al. 2022 or Song et al. 2020 and illustrated in Figure 3.4 :

— **Policy Contract (PC)** : defines and manages access control policies.

— **Policy Decision Contract (PDC)** : evaluates submitted requests against the policies.

— **Access Contract (AC)** : issues access tokens following successful validation.

This modular separation of responsibilities increases maintainability and auditability.

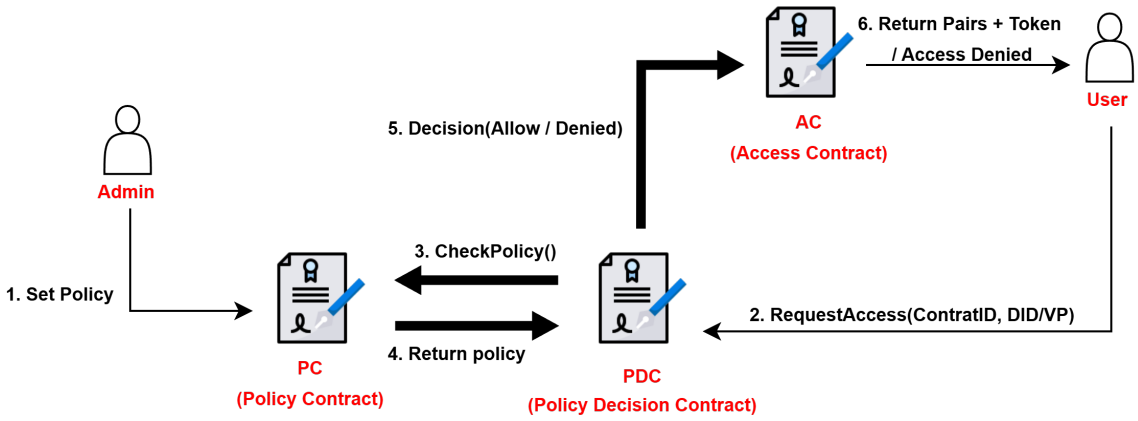

Figure 3.4 – Smart contract layer

The enforcement process of an access control policy is illustrated in the sequence diagram of Figure 3.5. A subcontractor initiates a request by submitting a VP and a ContractID. The PC checks the relevant policy, the PDC evaluates the request against attribute conditions and the AC issues a token if the request is allowed. All steps, including denials, are immutably logged on the blockchain.

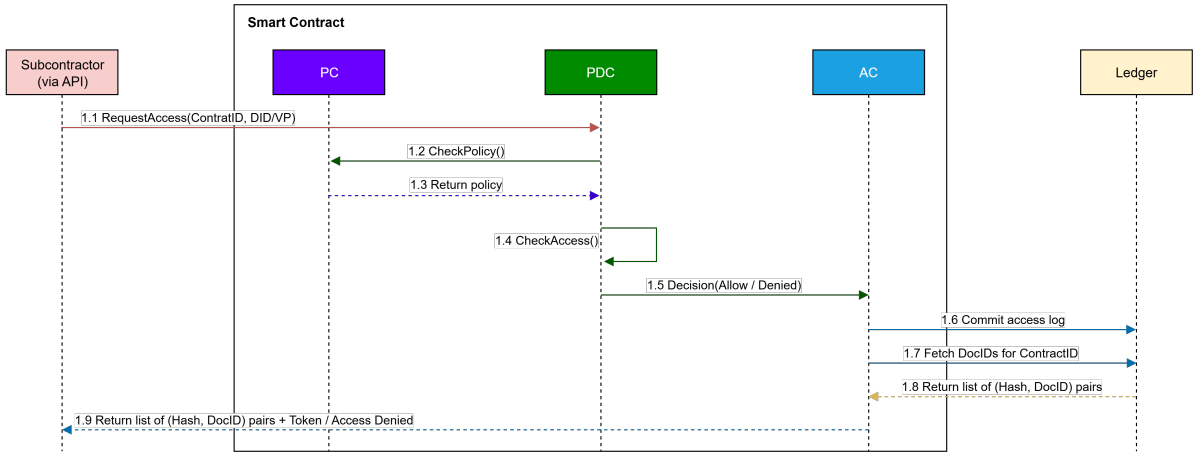

Figure 3.5 – Structure of the smart contracts layer

4 Implementation

This chapter details the concrete realization of the proposed model on a **Hyperledger Fabric** network, with **Veramo** for DID and VC/VP handling. We first outline the framework, then give technical details of the implementation.

4.1 Hyperledger Fabric

Hyperledger Fabric is a modular permissioned blockchain that replace order-execute with an **execute-order-validate** pipeline for higher throughput and parallel transactions. Fabric is an open-source framework to address common industrial needs, offering high modularity and flexibility as well as high performance. It provides identity management via **Membership Service Provider (MSP)** and X.509 CA, for participant or peers. **Smart contracts ("chaincode")** support, that can be written in various programming languages (GO, Java, JavaScript). And data scoping through **private channels** and **private data collections** to share data with only a subset of nodes of the blockchain system. It also adopts crash-fault and Byzantine Fault Tolerance (BFT) via **Raft** or **smartBFT** consensus algorithm. These properties align with our access-control objectives.

Fabric architecture is described as :

— **Orderers** : form the ordering service (Raft cluster) that sequences endorsed transactions into blocks. The outcome of this process is that it ensures that all transactions are recorded in the correct order.

— **Peers** : nodes that participate in transaction **validation** and **commit** blocks. They are also responsible for maintaining an immutable ledger and the state database. A peer may also act as an **endorsing peer** for a given chaincode, simulating proposals to produce read/write sets and endorsements under the chaincode's endorsement policy. **Leader** peers pull blocks from orderers for their organization, while **anchor** peers advertise endpoints to enable inter-organization gossip.

— **Membership Service Provider (MSP)** : the identity and trust layer of Fabric. It defines how an organization's member are identified and verified. Network and endorsement policies reference MSPs to authorize read, write and endorsement actions.

As illustrated in Figure 4.1, Fabric follows an execute-order-validate pipeline: during **Execute** endorsing peers simulate proposals and return endorsements; during **Order**, the ordering service batches endorsed transactions into blocks and delivers them to peers; during **Validate**, peer verify endorsements and discard invalid or conflicting transactions; finally, **Update state** commits valid transactions to the ledger and world state on all peers.

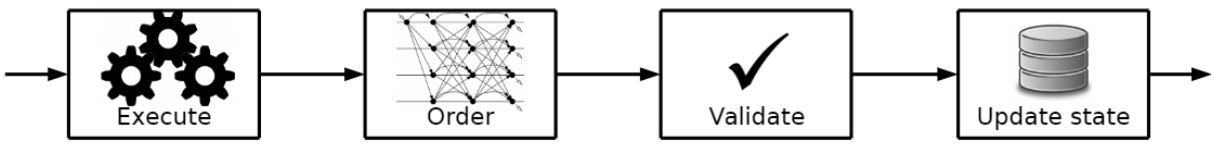

Figure 4.1 – Execute-order-validate architecture of Fabric, Androulaki et al. 2018

The deployment of the Fabric network uses Docker. Each component runs in its own container (orderers, peers, Fabric CA). Chaincode is executed in dedicated containers attached to endorsing peers.

4.2 Veramo

The identity layer uses Veramo to issue and verify VCs and assemble VPs. Veramo meets our needs in terms of the DID framework, as it natively handles VC and VP. Other frameworks were also considered, such as Hyperledger Aries but we chose Veramo to make the implementation easier. It provides key management, credential issuance, presentation assembly and verification. It creates JWT-based credentials (VCs) for attributes such as role and organization, binds them into VPs for specific requests, and verifies signatures and status.

4.3 Prototype implementation

The network comprises two organizations, Manufacturer (Org1) and Subcontractor (Org2), on a single channel with a Raft ordering service. Each organization operates a Fabric CA for enrollment and TLS, and at least one peer. Deployment relies on **Docker** : every component (orderers, peers, CAs) run in its own container, chaincode executes in isolated containers attached to endorsing peers.

Chaincode is written in Java using the Fabric Contract API. We follow the Fabric chaincode life cycle :

1. **Package the chaincode :** This step is completed by one organization
2. **Install the chaincode on peers :** Every organization that will use the chaincode to endorse a transaction or query the ledger needs to complete this step.
3. **Approve for each org :** Every organization that will use the chaincode needs to complete this step. The chaincode definition needs to be approved by a sufficient number of organizations to satisfy the channel's LifecycleEndorsement policy (a majority, by default) before the chaincode can be started on the channel.
4. **Commit the chaincode definition to the channel :**One organization submits the commit once the required approvals exist. The submitter collects endorsements from enough peers of the organizations that have approved, and then submits the transaction to commit the chaincode definition.

The Manufacturer deploys a Policy Contract to persist ABAC policies, and a Policy Decision Contract (PDC) that exposes checkAccess and records a permit/deny decision and metadata on-chain. Events are emitted on each decision for tamper-logging.

DID and VC/VP are generated with Veramo. The manufacturer issues JWT-based VC that encode requester attributes. The subcontractor assembles a VP that binds these claims to access request, including a ContractID. In the current prototype, end-to-end VP verification is performed off-chain and is not yet wired into Fabric. The chaincode simulates VP handling instead. The PC stores the manufacturer's policy and the PDC accepts a JSON payload with attributes and the ContractID, evaluates the request against the on-chain policy, records the decision and emits an event. This design lets us check that our model logic works without having to fully integrate all the elements mentioned. The next section outlines the priority missing pieces.

4.4 Future Work

Due to time constraints, not all planned features and components were fully implemented such as :

— **End-to-end VC/CP :** replace the VP simulation in chaincode with VP full integration.
— **Access tokens :** replace the placeholder token with signed, single-use token bound to the ContractID and requester DID.
— **Privacy :** protect attributes by moving decision inputs to private data collections so subject claims do not appear on the public ledger.
— **Organizations :** extend the consortium with a many organizations and certifiers.

There is still time before the end of the internship, the next iteration will prioritize wiring VP verification and integrating the model into the operational environment.

Conclusion

This work studied how blockchain can support distributed access control in multi-cloud environment and proposed a concrete, auditable ABAC model on Hyperledger Fabric. The model externalizes policy to smart contract, binds requests to decentralized identifier and enables multi-cloud integrations. The prototype, still unfinished, reinforces the idea that this is achievable and even more.

The internship produced this report, including a structured state of the art, a theoretical model with formalization of access control on a realistic scenario and a functional implementation : a Fabric network, Java chaincode that stores policies and evaluates access, and an integration of decentralized identifier.

4.1 Acquired knowledge

Domain knowledge in IT security / cybersecurity

I deepened my understanding of access control models (ABAC, RBAC, MAC, DAC) and learned how to select the most appropriate one by adapting its strengths and weaknesses to the context. I analyzed blockchain security across public, private and permissioned networks, and why permissioned ledgers suit an enterprise context. I examined multi-cloud governance, cross domain policy interoperability, shared trust anchors and auditable coordination across organizations and how blockchain can provide this layer while preserving data sovereignty. I learned decentralized identifier basics and how they can replace existing solutions. I consolidated integrity and auditability concepts via hash-chained blocks, ordered consensus and immutable logs.

Cross-cutting competencies

I strengthened project management like prioritized tasks and defined the scope of the project. I improved literature review and how to synthesize state of art. I also improved scientific article writing, with clearer structure and precise technical vocabulary. I learned containerization with Docker and how to deploy networks.

Specific skills in terms of IT security / cybersecurity

I deployed a Hyperledger Fabric network end to end by configuring CAs, enabling, TLS and creating channels. I implemented and tested chaincode in Java. I executed DID workflows by issuing credentials, assembling presentations, verifying signatures and status.

Bibliography

[WG18] Karl Wüst and Arthur Gervais. "Do you Need a Blockchain?" In: *2018 Crypto Valley Conference on Blockchain Technology (CVCBT)*. 2018, pp. 45–54. DOI: 10.1109/CVCBT.2018.00011 (cit. on p. 6).

[NL25] Montassar Naghmouchi and Maryline Laurent. *Privacy by Design for Self-Sovereign Identity Systems: An in-depth Component Analysis completed by a Design Assistance Dashboard*. 2025. URL: https://arxiv.org/abs/2502.02520 (cit. on p. 7).

[Wij24] Patikiri Arachchige Don Shehan Wijesekara. "A Literature Review on Access Control in Networking Employing Blockchain". In: *Indonesian Journal of Computer Science* 13 (Feb. 2024). DOI: 10.33022/ijcs.v13i1.3764 (cit. on p. 9).

[Pun+24] Aarti Punia, Preeti Gulia, Nasib Singh Gill, et al. "A systematic review on blockchain-based access control systems in cloud environment". In: *Journal of Cloud Computing* 13.1 (Sept. 2024), p. 146. ISSN: 2192-113X. DOI: 10.1186/s13677-024-00697-7 (cit. on p. 9).

[Li+22] Qi Li, Zhen Yang, Xuanmei Qin, et al. "CBFF: A cloud–blockchain fusion framework ensuring data accountability for multi-cloud environments". In: *Journal of Systems Architecture* 124 (2022), p. 102436. ISSN: 1383-7621. DOI: https://doi.org/10.1016/j.sysarc.2022.102436 (cit. on p. 10).

[PSS24] Youngho Park, Su Shin, and Sang Shin. "Cross-Domain Bilateral Access Control on Blockchain-Cloud Based Data Trading System". In: *Computer Modeling in Engineering and Sciences* 141 (Aug. 2024), pp. 1–10. DOI: 10.32604/cmes.2024.052378 (cit. on p. 10).

[Liu+23] Yang Liu, Weidong Yang, Yanlin Wang, et al. "An access control model for data security sharing cross-domain in consortium blockchain". In: *IET Blockchain* 3.1 (2023), pp. 18–34. DOI: https://doi.org/10.1049/blc2.12022 (cit. on p. 10).

[Shi+22] Dong-Her Shih, Ting-Wei Wu, Ming-Hung Shih, et al. "Hyperledger Fabric Access Control for Industrial Internet of Things". In: *Applied Sciences* 12.6 (2022). ISSN: 2076-3417. DOI: 10.3390/app12063125 (cit. on pp. 11, 17).

[Son+20] Lihua Song, Mengchen Li, Zongke Zhu, et al. "Attribute-Based Access Control Using Smart Contracts for the Internet of Things". In: *Procedia Computer Science* 174 (2020). 2019 International Conference on Identification, Information and Knowledge in the Internet of Things, pp. 231–242. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2020.06.079 (cit. on p. 17).

[And+18] Elli Androulaki, Artem Barger, Vita Bortnikov, et al. "Hyperledger fabric: a distributed operating system for permissioned blockchains". In: *Proceedings of the Thirteenth EuroSys Conference*. ACM, Apr. 2018, pp. 1–15. DOI: 10.1145/3190508.3190538 (cit. on p. 19).