Budapest University of Technology and Economics
Faculty of Electrical Engineering and Informatics
Department of Measurement and Information Systems

Model-driven decision support based on privacy-aware blockchain smart contracts

Scientific Students' Association Report

Author:

Bence Benyák

Advisor:

dr. László Gönczy
Attila Klenik

Contents

Kivonat

Szervezetek együttműködése során kritikus kérdés a közösen használt rendszer üzemeltetése és a rendszerbe vetett bizalom. A blokklánc technológia lehetőséget kínál arra, hogy akár ellenérdekelt felek is úgy tudjanak együttműködni, hogy biztosítsák adataik védelmét és kizárják a visszaélés lehetőségét. Több szektorban is találkozhatunk olyan szervezetekkel vagy vállalatokkal, amelyek működéséhez a kollaboráció elengedhetetlen. Az együttműködés szabályait leírhatjuk üzleti folyamatként, illetve ezekbe ágyazott üzleti szabályrendszer formájában is.

Szabály alapú döntések modellezésének szabványos nyelve a DMN (Decision Model and Notation), az Object Management Group által elfogadott szabvány, melynek segítségével üzleti szakértők számára is olvasható, hierarchikus, közvetlenül kiértékelhető, üzleti folyamatokba ágyazható szabályrendszer készíthető. Ezeket a döntési táblákat alkalmazhatjuk többek között adatok érvényesítésének vizsgálatára, csoportosításra, kalkulációra és komplex döntések szabványos leírására. Elosztott üzleti együttműködésekben ugyanakkor gyakori, hogy a döntési logika és maguk a döntési faktorok bizalmasak, azokat az egyes szereplők nem akarják felfedni. A munkám során üzleti szabályrendszerekből állítok elő automatikusan olyan blokklánc alapú alkalmazásokat, melyek figyelembe veszik az adatrejtés (privacy) szempontjait is. Munkámat digitális jegybankpénz (Central Bank Digital Currency) kezelését bemutató esettanulmányon keresztül ismertetem.

Vállalatközi blokklánc rendszerek megvalósításához az egyik leggyakrabban alkalmazott technológia a Hyperledger Fabric privát blokklánc keretrendszer, amely az ipari felhasználási esetek széles körét támogatja. A Fabric paraméterezhető hálózati és együttműködési mechanizmusokat nyújt és testreszabható adatrejtési módszereket támogat, így a blokklánc csomópontjainak csak egy része látja a döntélhozatalban felhasznált adatokat, de mindegyik eltárolja azok hash értékét. Kettő okos szerződést implementáltam munkám során, amelyek kezelik a blokkláncon tárolt információkat és elválasztják az üzleti logikát és az adatok hozzáférésének szabályozását, tetszőleges számú szabályrendszerre. Az egyik szerződés üzleti szabály kiértékelő motor funkcionalitást tölt be, a kiértékelendő szabályrendszerből automatikusan előállított adatstruktúra kódolja el a döntési logikát, a végrehajtási szemantika a DMN által definiált szabályillesztési módokat követi. A döntési folyamat során az általam elkészített hozzáférést vezérlő okos szerződés felelős a szerepalapú hozzáférési sémá betartásáért. A jogosultságok felvétele a szabályrendszer eltárolásával egyidőben, automatikusan történik. A feltöltést megelőzően érdemes a szabályrendszeren megfelelősségi ellenőrzést futtatni, hogy a blokkláncon minden esetben a kritériumoknak megfelelő szabályrendszert tudjunk eltárolni, a munkámban erre is adok megoldást. Egyedi adatmodellt alkalmaztam a szabályrendszer tárolására, továbbá a kiértékelés folyamatának követésére, ezzel elősegítve az adatvédelem és a végrehajtás nyomonkövethetőségének szempontjait. Az általam elkészített módszer bemutatja hogyan használhatjuk ki egy privát blokklánc előnyeit egy részszámításokra bontható probléma esetén.

Abstract

In case of business collaborations, trust and governance of underlying IT systems are key issues. Blockchain technology offers an opportunity to set up systems without one central trusted entity, with guarantees on data integrity. In many business domains, organizations or companies need to collaborate in their operations. Policies of such collaborations may be described as business processes or a set of business rules which support decision-making.

Rule-based decisions can be captured by the standard Decision Model and Notation (DMN) by the Object Management Group. DMN provides a way to create a universally readable, hierarchical model from a set of rules and embed it into business processes as needed. Decision tables can be used for data validation, clustering, calculation, and other business rules creation in a standard way. A typical case is that the decision logic and decision factors are confidential information in the company's life, but without them, no common decision can be made. In my work, I automatically generate blockchain-based applications from business rule sets with privacy. My work is presented through a case study of central bank digital currency management.

One of the most commonly used technologies for implementing enterprise-wide blockchain systems is the Hyperledger Fabric private blockchain framework, which supports a wide range of industrial use cases. Fabric provides parameterizable networking and collaboration mechanisms and supports customizable privacy methods so that only a subset of the nodes in the blockchain can see the data used in the decision-making, but each of the parties stores its hash value. I have implemented two smart contracts to manage the information stored on the blockchain and separate the business logic from the data access control for any number of rulesets. One of the contracts functions as a business rule evaluation engine, the data structure is automatically generated from the rule set, and the execution semantics follows the rule fitting methods defined by the DMN. During the decision-making process, the smart contract I have implemented to control access is responsible for enforcing the role-based access schema. Adding permissions is done automatically at the same time as storing the ruleset. Before uploading it is worth running a compliance check on the ruleset to ensure that the ruleset is always stored on the blockchain according to the criteria, I provide a solution for this in my work. I have used a custom data model to store the ruleset and to track the evaluation process, thus facilitating privacy and traceability aspects of execution. The suggested method for distributed rule evaluation demonstrates the benefits of the private blockchain for decision-making that can be executed as partial computations.

Chapter 1

Introduction

In the first chapter, I would like to give a general background to my work and position it according to its applicability in the real world. I present related works and I describe how I can offer more than these with my solution. To demonstrate my method, I will also give a case study, which will be used throughout the paper. I will explain the technical details in later chapters, the purpose of the introduction is to share my motivation with the reader.

1.1 Business orchestration

For a larger company, organizing the company's operations is essential. There are many representational models of how a company works, which can be used to create transparent models that everyone can read. Business users also create executable models to automate tasks of their company. A common solution in the business sector today is to use a system that can provide different functions in the operation. In such systems, meetings are held, work days are scheduled, work processes are assigned, monitoring, payments and invoicing are often also part of the system's responsibilities.

Enterprise Resource Planning (ERP) refers to a type of software that organizations use to manage day-to-day business activities such as accounting, procurement, risk management, and compliance, project management, and supply chain operations. One example is SAP,[1] the world's leading ERP software vendor. This type of system is often third-party managed, especially when several collaborating organizations need to be organized on a common platform. Another approach is to ensure that all parties are part of a common system, such as TradeLens,[2] which is an open and neutral supply chain platform underpinned by blockchain technology.

My ambition is to create a distributed system, where participants are part of the system and can perform transparent business management. In my work, I focus on two modeling

[1]Available here:`https://www.sap.com/about.html`
[2]Available here: `https://www.tradelens.com/`

languages, Business Modeling and Notation (BPMN), which is a graphical representation for describing business processes, and Decision Model and Notation (DMN), which is a powerful tool for describing business rules.

1.1.1 Business Process execution over blockchain

We can see processes as algorithms, which determine how an organization runs. These processes can be modeled and several software applications offer the opportunity to execute them. When we create a BPMN we have a tool to group our tasks by participants. It can be used for separate software layers, fields in companies or different cooperative organizations. Collaborative business processes are especially trust-intensive with mutually un-trusted companies. The blockchain technologies have the potential to innovate how organizations operate. Numerous research proposals have demonstrated the feasibility of designing blockchain-based collaborative business processes using a high-level notation, such as BPMN, and thereon automatically generating the code artifacts required to execute these processes on a blockchain platform [17].

BPMN offers the possibility to embed DMN into the process. This can also be done on the blockchain, if you have a ready-made BPMN engine and DMN engine that can operate over the blockchain. The modulation property of the technology allows the two engines to work together. This enables organizations to engage in common processes and decision-making.

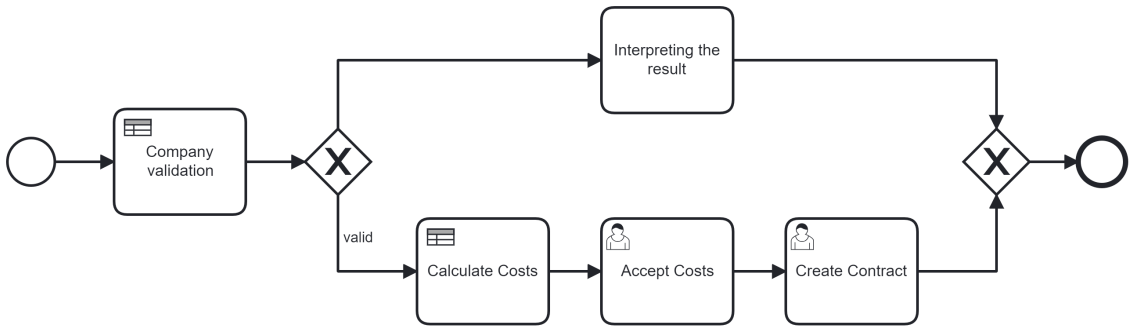


Figure 1.1: Example BPMN with embed DMN from the case study.

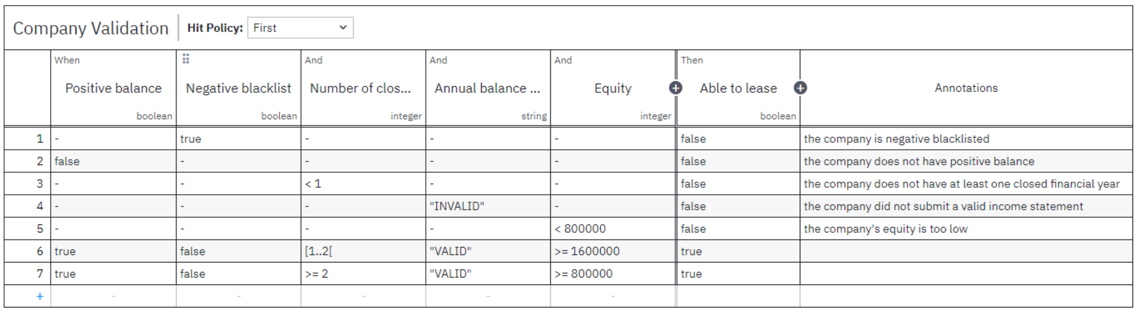
1.1.2 Business rule management

There are a number of decisions in the operation of the company. These can be based on written and unwritten rules. By setting these rules and basing our operations on them, we can increase the efficiency of our company. The decisions become transparent and readable and we are able to automate them. The fact that the logic of our decisions is standardized brings additional benefits. It makes it easier to share them with our business partners and reduces the chance of inconsistency. We can run different algorithms on the ruleset to detect gaps or inconsistencies in the business logic. To execute a set of rules, we need to take several steps; we need to create and maintain them and need an execution engine to make automatic decisions. We would like to perform these steps in one system, and BRMS provides the opportunity to achieve this.

Definition 1. A Business Rule Management System (BRMS) defines, deploys, executes, monitors, and manages business rules and decision logic. The system automates rules and decision-making across processes. [12]. ▪

Using a business rules engine, BRMS acts as a central repository of business rules and automates operations. Decision-makers and IT professionals can collaborate to create, copy and edit rules in a common environment.

To describe the ruleset we need a modeling language, I chose DMN, which has many beneficial properties. By way of mention, it allows business rules to be defined in a standardized format. More information about the standard and its advantages can be found in Section 3.2. The figure below shows an example decision table from the case study, where the columns are the different decision factors and outputs, and the rules are defined in the rows.



Company Validation | **Hit Policy:** First

	When		And	And	And	Then	
	Positive balance	Negative blacklist	Number of clos...	Annual balance ...	Equity	Able to lease	Annotations
	boolean	boolean	integer	string	integer	boolean	
1	-	true	-	-	-	false	the company is negative blacklisted
2	false	-	-	-	-	false	the company does not have positive balance
3	-	-	< 1	-	-	false	the company does not have at least one closed financial year
4	-	-	-	"INVALID"	-	false	the company did not submit a valid income statement
5	-	-	-	-	< 800000	false	the company's equity is too low
6	true	false	[1..2[	"VALID"	>= 1600000	true	
7	true	false	>= 2	"VALID"	>= 800000	true	
+	-	-	-	-	-		

Figure 1.2: Company validation decision table from the case study, complete version.

1.2 Blockchain for cooperative organizations

Blockchain technology has become familiar to everyone in today's world through Bitcoin. Alongside the emergence of cryptocurrencies, blockchain technology has also made advances in the scientific world. Take, for example, the potential of digital money, a number of studies analyzing the feasibility of Central Bank Digital currencies (CBDCs). In fact, there are already several real-world implementations of CBDCs, for example, China's new digital currency, the e-CNY, is being used to make 2 million yuan ($315,761) or more of payments a day in its latest pilot at the Beijing Winter Olympics, a top official from the Chinese central bank (PBOC) has said.[3] These developments also show that we can expect to see more blockchain-based solutions in the future. For those more experienced with the technology, it is evident that the logic of digital currency transactions is defined by smart contracts.

[3]Available here: `https://t.co/gE67fLL3d9`

These contracts are not only able to execute financial transactions. The company's business logic could also be applied to the blockchain, and it is suitable for validation, verification, and auditing. Also, inter-organizational cooperation is based on contracts, some of these contracts can be formulated as program codes. This allows mutually un-trusted parties to work together. A smart contract is accepted by all and furthermore run by all. This leads to tamper-proof, traceable collaboration.

In this paper, I describe the types of blockchains and their general structure. I present a couple of blockchain applications for business collaboration. And among the private blockchain technologies, I highlight HyperLedger Fabric, which assisted my privacy-preserving solution.

1.3 Sensitive data privacy and protection

Sensitive data is confidential information that must be kept safe and out of reach from all outsiders unless they have permission to access it. Access to sensitive data must be controlled through data protection and information security procedures designed to prevent data leakage and data security breaches.

Examples of this data are:

- Customer Information,
- Education Records,
- Card Holder Data,
- Personal Data,
- Business secrets,
- Confidential Information.

In the foreground of the technology I used, blockchain achieves decentralized security and trust in plenty of ways. It can be integrated into multiple areas; cryptocurrencies have already become known around the world, and we can find huge amounts of articles about them. It became general knowledge that cryptocurrencies are anonymous, but all transactions are accessible to anyone.

In our case, the main task was to hide certain data from the parties involved in the decision and to achieve full privacy. It is often that we have to release information that is a business secret in order to make common decisions. Another possibility is that we do not want to share the set of rules with all parties involved in the decision-making process or who will receive the results. And these two options can exist at the same time. For example from the case study, the statement of a company's bank account or the method used by the leasing dealer to calculate the lease fee for a car. Because this may be responsible for calculating the amount of the financial transfer, but the company does not want to give this information to the leasing company. It is important that this data is not only protected from the average user but also cannot be accessed by an IT expert.

Access control is a method of restricting access to sensitive data. Only those who have verified their identity can access data through the access control gateway. There are several types of data protection, about how we control access. Of these, I chose Role-Based Access Control (RBAC), as this paradigm fits the problem I was working on. In RBAC, roles represent functions within a given organization; authorizations are granted to roles instead of to single users. Authorizations granted to a role are strictly related to the data that are needed by a user in order to exercise the functions of the role [16]. In my case, I control the access to the business functions, and sensitive information according to the role of the participant in the decision-making.

1.4 Problem definition and motivation

I define the problem from two approaches, first I describe it in terms of decision-making, and then I define a general problem that can be solved based on my solution. Throughout the design, I took into consideration that the solution could also provide a model to match the requirements of the general problem. This means that my work not only provides a new solution for collaborative decision-making between cooperating enterprises but also it helps to create privacy-preserving platforms inspired by my architecture, in many fields.

1.4.1 Distributed, privacy-preserving decision-making, with a model-driven approach

1.4.1.1 Motivation

I chose a problem that I couldn't find a solution to in the existing works. I also took into account its usability, and based on these, I chose the creation of a BRMS as my project. Which is suitable for making decisions where different columns (inputs and outputs) are assigned to different entities. In addition to its wide use, the DMN can also be used to embed the decision task in business processes. Similarly, I offered a DMN engine over blockchain and my system will be also embeddable with the possibilities offered by blockchain technology (bridging, cross-chaincode calls).

1.4.1.2 Definition

I built a BRMS that offers complete privacy. This means the following:

- An organization can participate in decision-making without knowing the decision logic.
- The decision can be made without knowing the exact value of the inputs.

- The participant with the input value may not necessarily reach the output.
- The factors provided for the decision are stored encrypted in the system, so if a conflict needs to be resolved, the input provider can verify the correctness of the data.
- The result of the decision should be achievable without knowing the decision factors or the logic of the decision.
- Ensure that the decision is based on the set of rules we have agreed upon and the inputs we have provided.

1.4.2 Distributed execution of a calculation that can be partitioned into sub-calculations

1.4.2.1 Motivation and definition

My solution offers the possibility to perform calculations that can be split into sub-calculations. The sub-calculations belong to the participants, who perform these only in their own environment and deliver the results to those who are authorized to receive them. The data used for the calculation is protected, although it is kept in storage where we can publicly track any changes to it. This also assures the readers of the output that the calculated value was based on this value. Participants do not need to know the logic of the calculation, they can participate in the common calculation without knowing it. In summary, each participant knows only their own values, but they can perform a collaborative calculation.

1.5 State of the Art

Recently, blockchain technology has received a lot of attention from academia and industry. Interactions between businesses are based on a contract. The technologies that enable the execution of such contracts offer the opportunity to automate collaboration and shared decision-making.

Such work is for example the following article [23], which discusses the potential of using blockchain to implement business processes. Read the following in the article: emerging blockchain technology has the potential to drastically change the environment in which inter-organizational processes are able to operate. Blockchains offer a way to execute processes in a trustworthy manner even in a network without any mutual trust between nodes.

The benefits of the technology are demonstrated by the fact that alongside the implementation of business processes on the blockchain, there are a number of solutions that support enterprise collaboration. For example, FalconDB, which enables different parties with limited hardware resources to efficiently and securely collaborate on a database [24], and security guarantees at the blockchain level. Of course, collaboration is not just for companies, it is also present in governments and even among robots. Management and decentralising multi-robots have played a vital role in the fight against COVID-19 by reducing human interaction, monitoring and delivery of goods. Blockchain technology can decentralise the way multi-robots collaborate, improve the interaction between them to exchange information, share representation, share goals and build trust [14].

Decision-making can be done without blockchain, and there are many solutions for it, but none of them meet our privacy requirements. One example is the Camunda engine, which is designed for highly efficient execution of complex decision tables, supporting high throughput and requiring minimal infrastructure resources [6]. Thus, it fulfills many extra-functional requirements, but cannot support the common operation of un-trusted companies.

In [21] the developers presented a service, which is deployed on the Ethereum blockchain making data, logic, and execution transparent and tamper-proof. But this work has a significant limitation: public blockchains are transparent because all data and logic is publicly stored; therefore, they cannot be applied to privacy-sensitive data. The presented work lists two possible solutions: privacy-preserving blockchains, such as HAWK, and private blockchains that keep data and logic hidden.

At the end, I present the solution that comes closest to a completely privacy-preserving solution. In this article [20], they use a commitment scheme, which allow participants to publicly commit to secret data without revealing it. Input values are hashed onto the blockchain and it is possible to perform decision-making on the blockchain. The basic concept is that the outcome of the decision is calculated locally by each participant so that the value of each input is known. And if participants disagree on the output, a decision is made on the blockchain. But then they are not put on the blockchain as hashed values. There are two phrases in this work, the operational phrase, and the conflict resolution phrase. In my work, I combined the two and took advantage of the private blockchain to create a completely new solution.

I would also like to introduce a system that works in a real environment, the TradeLens platform mentioned before. This platform is able to achieve a certain level of privacy and one of its main purposes is decision-making. TradeLens is already handling more than 700 million events and 6 million documents a year, expediting decision-making and lowering the administrative frictions in trade.[4] I will explain the mechanism and their data protection system later, in Section 3.6.

1.6 Possible areas of use

In this section, I would like to present the areas where the use of a distributed BMRS can be beneficial. In addition to the business collaboration mentioned above, it can also be used within a company. In larger companies, it is common to divide the company into

[4]Available here: `https://www.tradelens.com/network`

departments. There is a weak relationship between them, only certain information needs to be exchanged. Within a department, you may come across data that you do not want to share with other departments. This could be, for example, the salary of employees, which we do not want to make public, but the calculation of the salary rate could be done on a decision table, where we need information from several departments. For example, an estimate from the finance department, the number of working hours from the human resources department, or feedback from a manager.

I would also mention the area of governance, which is where most personal data occurs. In some cases, in order to get a service, these government organizations decide whether we are valid for it. For example, taking out a loan, registering a car, applying for a building permit, etc. With the system I have created, it is possible for a bank offering credit to decide on your creditworthiness without having all the information about you. And the bank providing the loan does not have to share its terms with different organizations. Government blockchain applications are diverse in nature and include digital identity, the storing of judicial decisions, financing of school buildings and tracing money, marital status, e-voting, business licenses, passports, criminal records and even tax records [33].

Hiding the decision logic is particularly important for companies. A set of rules may contain their business logic or part of it, the sharing of which could be considered as sharing their business secrets. Thus, we can make use of it in B2B (Business-to-Business) processes. And it also gives the opportunity for cooperation where government departments can work with companies.

You can browse the websites of the HyperLedger[5] or the IBM[6] to find many more applications. Most of these are technologically close to my project and the execution of complex decision tables in these areas is essential. These are briefly listed as follows: supply chain, healthcare, retail, media and advertising, oil and gas, telecommunications, manufacturing, insurance, financial services, travel and transportation.

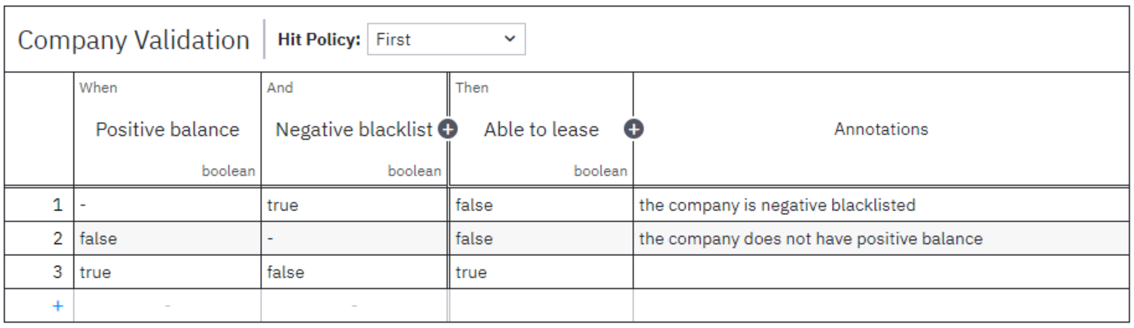
[5]Available here: `https://www.hyperledger.org/learn/case-studies`

[6]Available here: `https://www.ibm.com/blockchain`

1.7 Case study from financial sector

I would like to guide this work through a case study. Last semester I was a member of a team at the University, where we collaborated with the Hungarian National Bank. The finished work is a leasing process management and audit by blockchain technologies. In this work, we met lot of case, where we could use decision tables. During the process the payments were executed by a CBDC smart contract over HyperLedger Fabric network. One of the biggest advantages of the CBDC systems are that they are programmable and we would like to utilize it. The final work contains an extra function, which was colored CBDC. If you chose a "green-car" to lease, you would receive green money as state aid. We used a sample logic for this scenario, if the car was "green", you received half the price of the car in green CBDC.

We can see for this as a rule, but in a real system there should be more rules to calculate the amount of the state aid. This is one of the possible case, where we can use blockchain-based business rule execution. But in this work I tried to find the best example, which is also part of this leasing process and various organization participate in the decision. When a company want to leasing a car, several conditions must be met. I focused on company validation, where we can create several different decision table for this step. These conditions depend on the leasing company. I created a complete version decision table for company validation Figure 1.2, and a smaller version (1.3) from this for illustration in later chapters.


Company Validation — Hit Policy: First

	When Positive balance boolean	And Negative blacklist boolean	Then Able to lease boolean	Annotations
1	-	true	false	the company is negative blacklisted
2	false	-	false	the company does not have positive balance
3	true	false	true	
+	-	-		

Figure 1.3: Company validation decision table from the case study, sample version.

I created these tables using Camunda Modeler, which also allows you to create Decision Requirements Diagram (DRD). This is an advantage for us because it allows us to define a table with existing modeling elements where the columns belong to different organizations. As you can see in the example DRD (Figure 1.4), I associate input sources and knowledge sources to the decision table, which define the inputs and outputs in the table. Then, I link the identifiers of the organizations to these elements, achieving that we have all the information from the model description for the access control of the table.

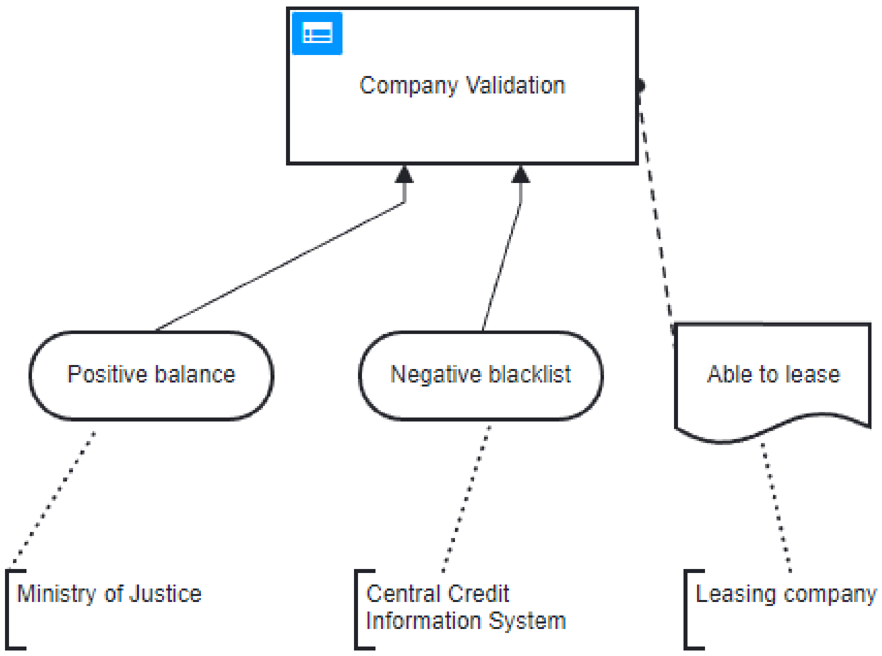


Figure 1.4: Decision Requirements Diagram (DRD) of Validation sample.

1.8 Structure of the document

This paper is organized as follows: in Chapter 2, I define a set of requirements to achieve complete privacy in the decision-making. In this chapter, I will also present the basics and challenges of access control and select a method to implement. The Chapter 3 explains the technologies in general and discusses the details that enabled the work to complete these requirements.

My development work is described in Chapter 4, presented from multiple viewpoints and levels of abstraction, and illustrated with figures, focusing on the results I have achieved. Furthermore, using the case study, I show how the final solution satisfies the requirements during a decision-making process. Finally, in Chapter 5 I conclude my results and discuss future work and potential improvements to my solution.

Chapter 2

Challenges and requirements

In this chapter, I define a set of requirements for extra-functional requirements and provide both proof and solutions to satisfy them in later chapters. These requirements illustrate why my solution offers more than previous implementations and why it has the potential for wider use. This chapter describes the options, models and methods used for access control. I present my chosen model and its challenges.

2.1 Extra-functional requirements

Privacy and Security

R.1 privacy and security: we want our data to be protected and secure, safe from attacks and leaks.

Immutability and Integrity

R.1.1 immutability and integrity: we want to make it impossible for someone else to make changes to our sensitive data. In this way, the system cannot be manipulated and data cannot be tampered with.
R.1.1.1 the uploaded ruleset cannot be edited by anyone.
R.1.1.2 the value of the input provided for decision-making cannot be modified, nor by the data provider.
R.1.1.3 the result of the partial calculations cannot be modified.

Traceability

R.1.2 traceability: we want the decision-making process to be traceable to the decision-makers without revealing protected data.
R.1.2.1 the decision-making process must always have a status that describes the current state of the process.

R.1.2.2 the execution and output of the completed decision-making process must be traceable, i.e. there should be a way to track how decisions were made. In the case of a conflict, the conflict should be resolved.
R.1.2.3 sensitive data changes must be traceable, in which case we want to know the modifier ID and a timestamp.

Transparency

R.1.3 transparency: the program must offer the possibility to check the correctness of the data.
R.1.3.1 the ruleset must be verifiable by participants who have a definition of it.
R.1.3.2 the result of the sub-calculations can be verified with the stored input value.

Permissioned

R.1.4 permissioned: access to data must be controlled.
R.1.4.1 parts of the decision table must only be accessed by those authorized to access them.
R.1.4.2 the specified input value is only available for the provider.
R.1.4.3 the calculated output value is only available for those parties who are authorized.
R.1.4.4 sensitive data cannot be reached from the communication channels.

Efficiency

R.2 efficiency: we want an efficient, automated operation in which a non-IT person can participate.
R.2.1 decision asset creation is supported by the ruleset defined by the DMN standard, so without an extra layer, decisions can be made based on the readable standard.
R.2.2 the system must be able to handle multiple sets of rules.
R.2.3 the system must be able to handle multiple executions of a set of rules.

Automation

R.2.4 automation: many data can be derived from the uploaded DMN. We want to automate the storage of these.
R.2.4.1 the storage of the ruleset and the setting of permissions are automatically done from the DMN standard.
R.2.4.2 the permissions of the calculated values are automatically stored.
R.2.4.3 a decision is made automatically when its requirements are satisfied.

2.2 Role-Based Access Control

2.2.1 Access Control

Access control reduces the risk of unauthorized access to physical and computer systems and is essential to information security, data security, and network security.

Definition 2. Access control is the process of mediating every request to resources and data maintained by a system and determining whether the request should be granted or denied. The access control decision is enforced by a mechanism implementing regulations established by a security policy. [26]. ▪

2.2.1.1 Components of access control

There are five main components to any access control system, whether physical or logical:

- **Authentication:** the act of proving an assertion, such as the identity of a computer system user. In my system, this is guaranteed by the blockchain system with digital identification.
- **Authorization:** the ability to grant access rights or privileges to resources. This is guaranteed by a smart contract. This is one of the most important methods in my work, I offer a multi-layered solution to managing this.
- **Access:** after authentication and authorization, the person or computer can access the resource.
- **Manage:** access control management involves adding and removing authentication and authorization for users or systems. I do this management in part through automation.
- **Audit:** used as part of access control to enforce the principle of least privilege. I will not go into this in my work, it can be part of further development.

2.2.1.2 Models of access control

Access control models are used to decide on the ways in which the availability of resources in a system is managed and collective decisions of the nature of the environment are expressed [28]. In this subsection, I briefly describe the main models of access control used. In the design phase, I had to overview these and then choose the method that fit the problem I had defined.

- **Attribute-based (ABAC):** access is granted based on attributes instead of user rights after authentication. One example is age-based access. Our identity does not have any attributes, and we do not want to store this kind of data about users on the ledger. We can assign roles within the organization, and we can separate the organization into units, but in my solution, we need to assign privileges to the organizations.

- **Role-based (RBAC):** policies control access depending on the roles that users have within the system and on rules stating what accesses are allowed to users in given roles [25]. I used an alternative version of this in my solution.
- **Mandatory (MAC):** access rights are controlled by a central authority based on levels of security. In most use cases, there is no central party responsible for decision-making. In addition, we would be entrusting an organization with a heavy responsibility, which conflicts with my main concept.

2.2.2 Access Control in Collaborative Systems

We can define requirements for access control in a collaborative environment. They facilitate efficient and trustworthy collaboration, and transparency is important to ensure that all parties can be sure that their data is protected, without having direct control over access control management.

The requirements defined for access control are:

- The access control needs to be applied and enforced at the distributed platform level.
- Model for access control must be generic and must allow the configuration of access rights to meet a wide variety of user needs collaboration tasks and enterprise models.
- Access control model must be able to provide strong protection for different types of distributed environments and objects and allow highly sensitive control of the access to objects and their attributes.
- Access control model require greater scalability in terms of the number of operations than traditional single-user models because the number of shared operations is much greater in collaborative environments compared to traditional single-user systems.
- All participants should have an insight into the model used.
- Non-cooperating organizations cannot access the stored permissions.
- Access control models for collaboration must be dynamic, that is, it should be possible to specify and change policies at runtime depending on the environment or collaboration dynamics.

2.2.3 Role-Based Access Control

My approach is based on the granting of access rights according to the role of the organization in decision-making. I do not allocate the rights within the organization, the organizations have them and they are assigned to a subject. Future development could be the addition of rights management within the organization. RBAC provides fine-grained control, offering a simple, manageable approach to access management that is less error-prone than individually assigning permissions [4].

2.2.3.1 Benefits of RBAC

Possibility to:

- Create automated, repeatable allocation of rights.
- Checking user privileges and correcting identified problems.
- By restricting access to sensitive information, you can reduce the risk of data breaches and data leakage.
- More effective compliance with regulatory and legal standards on confidentiality, integrity, availability, and data protection.

2.2.3.2 Access control list

I also used an RBAC alternative, Access Control List (ACL), which is a table, that contains permissions attached to subjects. I used a combination of the two methods to achieve the required result. Let's look at an example of such a list:

Subject	Organization identifier	Role
InputClause1_uwwxm2(Negative Blacklist)	CentralCreditInformationSystemMSP	addInput
DecisionTable_174e2xx(Company validation)	LeasingCompanyMSP	startProcess
Output_1(Able to lease)	LeasingCompanyMSP	getOutput
...	...	...

Table 2.1: Example ACL for Company Validation decision table.

It can be seen that the subjects are part of the decision table, and the identifiers of the organizations participating in the decision can be assigned roles, which are the names of the implemented functions.

Possible roles for the decision table assigned by the system:

- **startProcess:** It can start a decision-making process and access the information needed for the initiation. Access the identifiers of the organizations involved in the decision and the number of inputs.
- **addInput:** It has the right to give input when making a decision, for this, it accesses the input metadata. In addition, it accesses the values in the column only during the runtime of the calculation.
- **getOutput:** It has the right to reach output when a decision was made, and it accesses the output metadata. In addition, it accesses the hit policy, the values in the column, and the results of the sub-calculations only during the runtime of the calculation.

Chapter 3

Background

In this chapter, I describe the technologies used and the reasons behind my decision to use them. My work is to create a DMN engine over blockchain, hence the two technologies presented are DMN and blockchain. To understand my work, it is necessary to understand their basic structure, and for me, it was essential to discover their advantages, which would allow me to reach my solution.

3.1 Model-driven execution

Working in the blockchain is especially for people with technical skills. Organizations that want to move the coordination and monitoring of information sharing from their inter-organizational business processes to blockchain are hampered by technical skills requirements. Executives and business analysts should be connected through a model-based approach, reducing the need to know the details of the coding language to create, manage and monitor the smart contracts underlying collaborative processes. The design of smart contracts should be comprehensible, fast, reliable, and verifiable [17]. This is facilitated by Model-Driven Development (MDD).

One of the main features of MDD is that the focus of software development and product development is on models rather than computer programs. A key advantage of this is that the model is described in terms that are less closely related to the underlying implementation technology and closer to the problem area than most popular programming languages. This makes it easier to define, understand and maintain models; in some cases, systems can even be built by field experts rather than computer technologists. And the model is less sensitive to changes in the chosen computing technology. Consequently, a key core MDD principle is that programs are automatically generated from their corresponding models. Automation is by far the most effective technological means for boosting productivity and reliability [27].

My goal is to offer an environment that does not require programming knowledge to use. But we can implement rules using a high-level language. I want to take advantage of the possibilities offered by the MDD, including transparency, traceability, automation, and verifiability. During the design process, I thought about creating a code generator, but in my case, business logic and the data needed to execute it are both important knowledge and assets. So I suggest a model-driven solution where the code is given and can store the business logic as assets ready to be executed. Another advantage of this solution is that it provides the possibility of privacy.

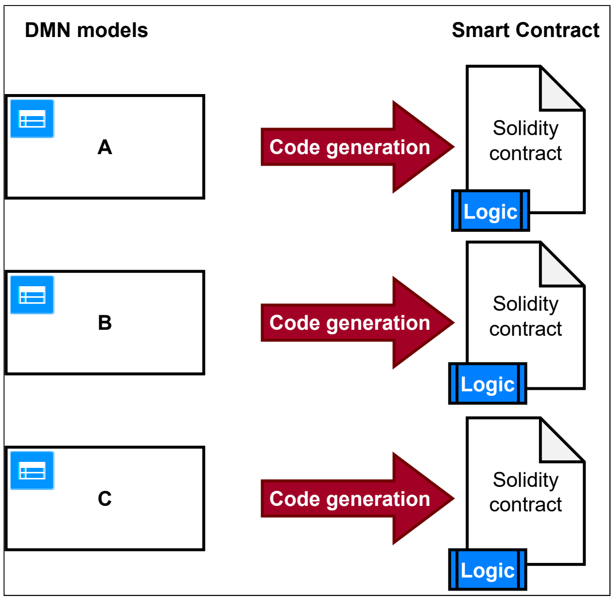


Figure 3.1: Generate solidity smart contracts from DMN models.

The two figures (3.1 and 3.2) clearly show the conceptual difference, which I will explain. The upper solution follows MDD principles more closely, as a code is automatically generated from the DMN model. The decision factors and decision logic are contained in the contract. The functions of the contract implement the evaluation of an expression. You can find an example of this solution [19], where you can read more about such a solution.

In my solution, the contract is fixed and all DMN models are handled by the same code. In this way, it is more like a DMN engine, which receives the model and evaluates the expressions using an evaluation engine. In the evaluation, the data is stored as assets, which are saved by the same contract. So there are various points at which the two approaches lead to different results. The first and most important is that the logic is saved in my solution by the smart contract from the XML, but in the above work, it is done by an off-chain application. You can also see that in my work, the decision logic is part of the assets.

The decision-making I offer over blockchain offers a lot more options for protecting and isolating data. As I will demonstrate in the description of the implementation.

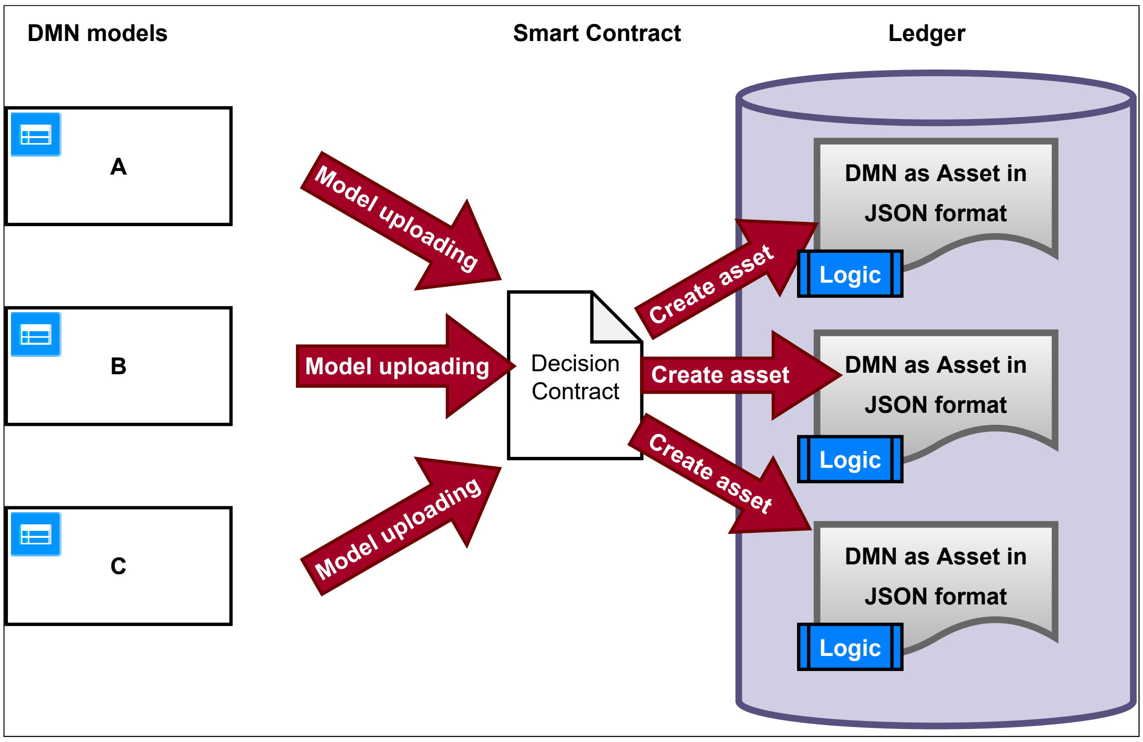


Figure 3.2: Parse DMN models to Assets on blockchain.

3.2 Decision Model and Notation

Definition 3. A **decision** is an act of determining an output value, based on a number of input values, using logic defining how the output is determined from the inputs. ▪

Definition 4. A **decision model** is a model in a specific area that defines decision requirements and decision logic. ▪

DMN is a modeling language and notation for the precise specification of business decisions and business rules, it is published by Object Management Group (OMG). DMN is easily readable by the different types of people involved in decision management [2]. It is built up of inputs and outputs, based on a when-then approach. In DMN, decisions are modeled and executed using the same language. It is designed to be embedded into CMMN and BPMN; Business Rule Task is a special kind of activity, which provides a mechanism for the process to provide input to a business rule engine and to get the output of calculations that the business rule engine might provide [1].

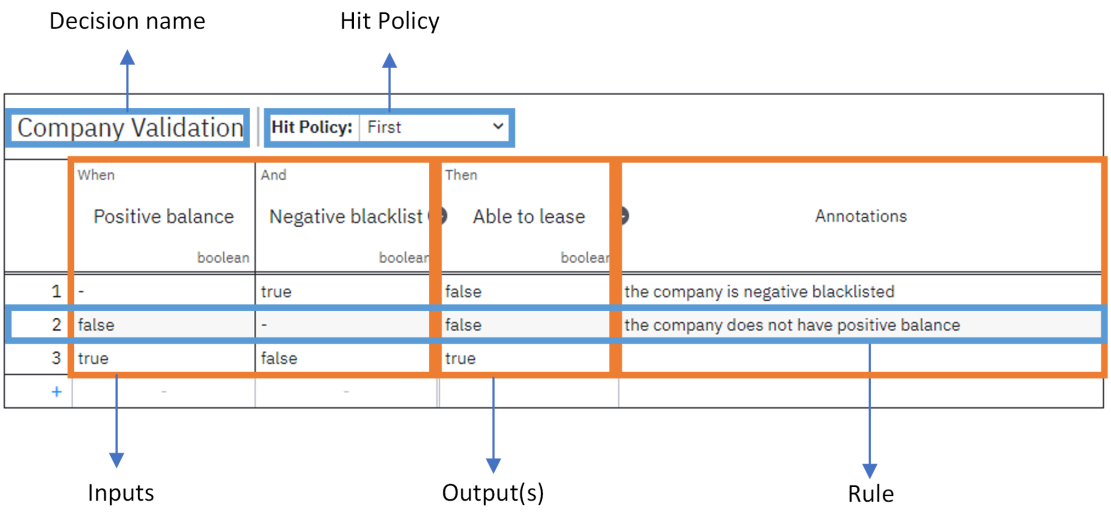
A DMN may contain a Decision Requirement Diagram (DRD), which provides meta-model, notation, and semantics for decision modeling. In DRD, we can create decisions, knowledge sources, business knowledge sources, and input data. I used two of these types to define the inputs and outputs. In addition, I took advantage of the fact that we can link TextAnnotation to them, thus setting the organizations for the columns.

These elements and the structure of the DMN allow it to be used for collaborative decisions. The fact that DMN is the standard managed by OMG further strengthens its reliability, we can also take advantage of the fact that we already have a number of tools to validate DMN to predict possible errors in decision logic/syntax. This standard provides a powerful model-based implementation for my project and also offers several tools for evaluation, on which I was able to build a model-driven, reliable solution for execution.

3.2.1 Structure of a decision table

The DMN table is composed of metadata and rules, these are:

- **Table name**
- **Table ID:** it must be unique in a business rule execution engine.
- **Hit policy:** hit policies describe different methods to evaluate the rules. In the next section, I explain in greater detail the different DMN hit policies.
- **Input columns:** we specify a name, an identifier, and the type, which determines how it will be evaluated. For text type, we can also specify predefined values.
- **Output columns:** we specify a name, an identifier, and the type.
- **Annotations column:** in this section, we can add comments for the rules, which may explain or further describe our decision.
- **Rules:** the set of box expressions represented as a row in the table. The expressions belong to specific columns and are evaluated according to their types. If all expressions in a rule match with given input values, then the rule is a fitting rule.



Company Validation

Hit Policy: First

	When Positive balance boolean	And Negative blacklist boolean	Then Able to lease boolean	Annotations
1	-	true	false	the company is negative blacklisted
2	false	-	false	the company does not have positive balance
3	true	false	true	
+	-	-		

Figure 3.3: Example DMN with names of its parts.

3.2.2 Hit policies

Definition 5. The **hit policy** specifies what the result of the decision table is in cases of overlapping rules, i.e., when more than one rule matches the input data [2] .

A decision table usually consists of several rules. By default conception of DMN, the rules do not overlap, but it has changed over time along with the needs. If the rules overlap, which means more than one rule matches the given set of input values, then the correct output is selected using the hit policy. It can also be used to check correctness at design time. Because my goal was to create a reliable BRMS, I had to take into consideration the different mechanisms. As they can reduce the number of possible errors in the operations. The types of hit policies can be classified into two groups. A single match table returns the output of only one row; a multi-matched table can return the output of multiple rows (or a function of the output, such as the sum of the values).

Single hit policies:

- **Unique:** there is no overlap, and all rules are disjunctive. Only one rule can match. This is the default value.
- **Any:** there may be overlaps, but all matching rules show equivalent output values for all outputs (ignoring the rule annotations), so any match can be used. If the output entries are not equal, then the matching ruleset is incorrect and the result is undefined.
- **Priority:** multiple rules can match, with different output values. This policy returns the corresponding rule with the highest output priority. Output priorities are arranged in a sorted list of output values in descending order.
- **First:** multiple (overlapping) rules can match, with different output records. The first match is returned (and the evaluation may stop) according to the order of the rule.

Multiple hit policies:

- **Output order:** returns all hits in decreasing output priority order. Output priorities are specified in the ordered list of output values in decreasing order of priority.
- **Rule order:** returns all hits in rule order.
- **Collect:** returns all results in any order. A function can be added to the outputs. Functions are:
  - Sum: the result of the decision table is the sum of all the outputs.
  - Min: the result of the decision table is the smallest value of all the outputs.
  - Max: the result of the decision table is the largest value of all the outputs.
  - Count: the result of the decision table is the number of outputs.

3.2.3 Column types and FEEL

The wide use of the decision table is also facilitated by the fact that a variety of data types can be used in decision-making. And different types of columns have different evaluation modes, which will be run by the FEEL engine. The following types can be used when defining the rules: number, string, boolean, time, and date.

DMN offers advanced concepts for checking input values, all decision logic is represented as boxed expressions. Part of the DMN standard is the Friendly Enough Expression Language (FEEL). FEEL defines a syntax to describe the conditions according to which input data should be evaluated. It has the following features:

- Side-effect free.
- Simple data model with numbers, dates, strings, lists, and contexts.
- Simple syntax designed for a wide audience.
- Three-valued logic (true, false, null).

3.3 Camunda Platform

I used the Camunda Platform when I worked with DMN and BPMN, which is a free and open-source decision automation platform. Camunda provides a scalable process automation and orchestration platform. It brings powerful execution engines for BPMN processes and DMN decisions, paired with tools for collaborative modeling, operations, and analytics [7]. Additionally, Camunda provides a FEEL engine, which can be integrated into Java projects, which became useful for me because in my solution the decision table is evaluated in a custom way, so I was not able to use the DMN engine.

Many companies using this platform to improve their performance or efficiency. These can be found on the Camunda site,[1] and we can see how necessary the development of model-driven solutions is. With the modeling tool, Camunda Modeler, we can create BPMN, DMN, and CMMM models, these editors can even be embedded in our application, and we can create plugins for Camunda Modeler. Thus, I could create a BRMS as a D-app that implements execution using a blockchain.

3.4 Ruleset verification with static analyses

BRMS ensures that the rules are treated consistently to avoid ambiguity. In our case, we need to handle inconsistency, redundancy, equivalence, and invalid rules. Alongside semantic verification, we also want to analyze the ruleset syntactically. We need to do these before deploying to make sure that the table we are evaluating doesn't run into errors. It is also important that the rules defined in the table are correct so that you always get the desired output for the given inputs. Bugs can be found using various static analysis tools, of which several implementations are already available for DMN models. The use of this method improves the reliability of the system and can be applied in different layers.

My work does not include making such a static analysis. However, I looked for tools that would meet all my needs. I have found two tools with different advantages, which I will briefly describe while explaining why we should use these tools.

[1]Available here: `https://camunda.com/case-studies/`

3.4.1 DMN Verification Tool published by Springer Nature

The tool presented in this report allows to analyze DMN models both on a logic level and on a DRD level, which is a current issue faced in practice [22]. The project's creators have created a complete verification tool, eliminating all possibilities for errors and even providing a solution to resolve them. A key advantage of the application is that it also performs analysis on the DRD, which is particularly important in our case, considering that it is not supported to deploy an incomplete DRD to the system. It is used through a web interface,[2] but is also available from program code. By calling the appropriate endpoint, we can send the XML description of the DMN to be validated in the request body and receive the verification results in the response body.

3.4.2 Red6 DMN-check library

This tool also verifies the DMN both at logic and DRD level.[3] The advantage of this work is that it can be used in six different ways. This gives the possibility to add it to different layers, even if the previous tool can complement the client application, this tool can be embedded in the smart contract. For me, the verification is done on the client side with this tool, and in case of an error, the DMN file is not uploaded. Of course, this can be bypassed with another client application, so if you want to make this check mandatory rather than just an option, it should be part of the smart contract.

3.5 Introduction to blockchain systems

As I mentioned in Section 1.2, blockchains can be an appropriate component of cooperation between companies. As well as its usability, I would like to describe its structure and general operation, to a level that makes my work and design decisions understandable, paying particular attention to the features that encourage privacy. Blockchains can operate as decentralized programmable platforms [31].

Definition 6 (Blockchain). A blockchain is a type of distributed ledger technology (DLT) that consists of growing list of records, called blocks, that are securely linked together using cryptography. Each block contains a cryptographic hash of the previous block, a timestamp, and transaction data. ▪

In order to submit transactions, the participant must have an identity, which must be verified, usually by means of a digital wallet containing an asymmetric key pair. Asymmetric-key cryptography provides the ability to verify that the user transferring value to another user is in possession of the private key capable of signing the transaction. Transactions are executed by smart contracts, which define public functions. These are generally responsible for modifying the contents of the ledger. A smart contract is a collection of code

[2]Available here: `http://dmn.fg-bks.uni-koblenz.de/?dmnurl=dmn`

[3]Available here: `https://github.com/red6/dmn-check`

and data that is deployed using cryptographically signed transactions on the blockchain network (e.g., Ethereum's smart contracts, Hyperledger Fabric's chaincode). The smart contract is executed by nodes within the blockchain network; all nodes that execute the smart contract must derive the same results from the execution, and the results of execution are recorded on the blockchain [32]. The ledger is a place to store data, save transactions, and store assets. Generally, everyone has a copy of this, which helps to ensure security.

Blockchains can be categorized from several angles, with different types having different uses. They can be categorized based on their permission model, which determines who can maintain them:

- **Permissionless:** Permissionless blockchain networks are decentralized ledger platforms open to anyone publishing blocks, without needing permission from any authority [32].
- **Permissioned:** users publishing blocks must be authorized by some authority.

Blockchain networks can be categorized according to data (ledger content) visibility as well.

- **Public blockchain network:** In a public ledger, the record of transactions is public and the consensus protocol is open to anybody.
- **Private blockchain network:** direct access to blockchain data is limited to predefined users. Only participants that are registered on the blockchain network can download the ledger.

3.6 Hyperledger Fabric and its benefits

Hyperledger is a global enterprise blockchain project that offers the necessary framework, standards, guidelines, and tools to build open-source blockchains and related applications for use across various industries [8]. The Hyperledger Foundation was started by Linux Foundation in 2015. Since then, it has received a number of contributions from IT companies to support the collaborative development of blockchain projects.

One of these projects is HyperLedger Fabric (HLF), which is an enterprise-grade, distributed ledger platform that offers modularity and versatility for a broad set of industry use cases. The modular architecture for Hyperledger Fabric accommodates the diversity of enterprise use cases through plug-and-play components, such as consensus, privacy, and membership services [9]. It is a permissioned, private blockchain that was designed for enterprise use.

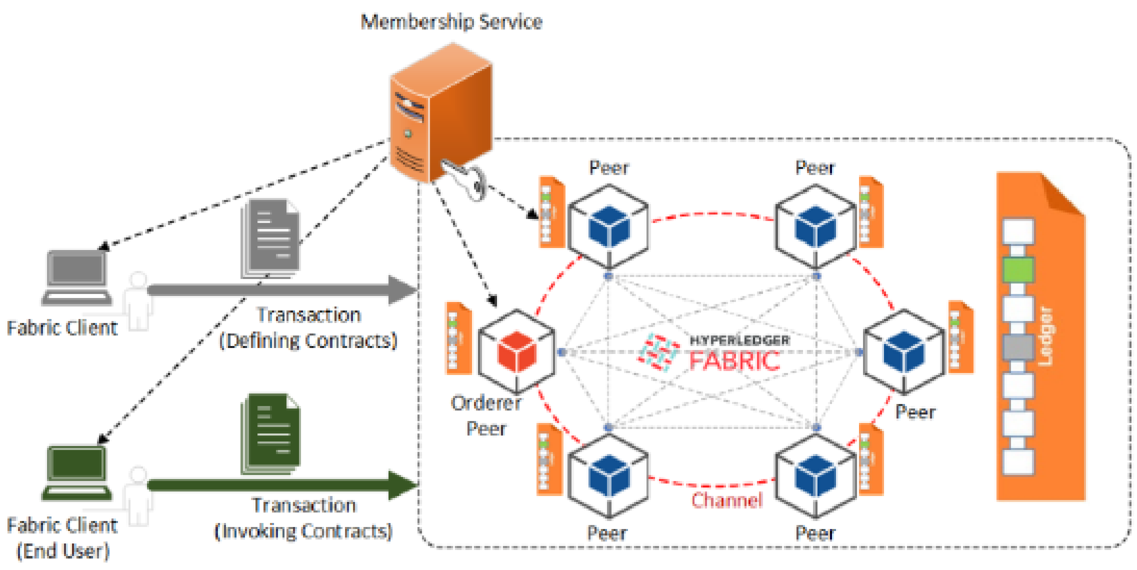


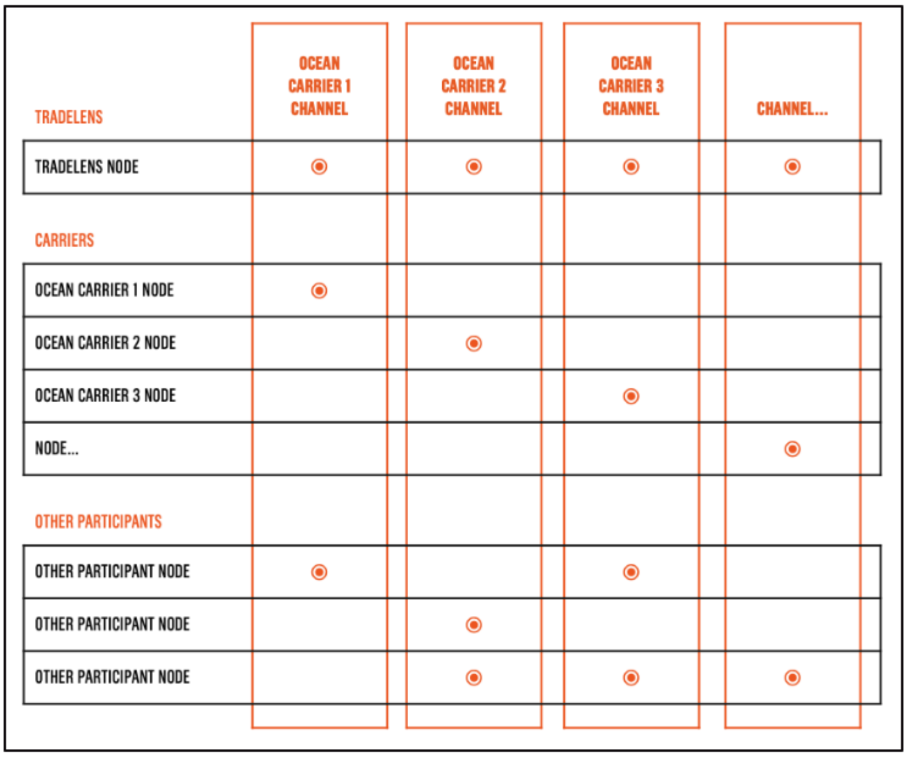
Figure 3.4: HyperLedger Fabric Architecture [15].

3.6.1 The architecture of the Hyperledger Fabric Network.

This figure shows the fabric structure for an end user. The figure shows that the client side is where the contracts are defined and called. In order to submit transactions, we have to pass an authorization process which is performed by the membership service. The Ordering Service performs the ordering of the accepted transactions. In a network organizations can be separated by creating channels, which enables different collaborations. Organizations have peers, chaincodes run on the peer and each peer has its ledger. We can upload a chaincode to a channel, which means that the peers on the channel will run it. Each chaincode has its own ledger on the peer and a chaincode can define multiple contracts. The chaincode definition can also include the specification of private data collections. The chaincodes run in docker containers, which help to separate them from each other. The main library is implemented with either CouchDB or LevelDB, which are lightweight libraries for building a key-value data store.

There are several ways to achieve privacy, for access control and data isolation, the Fabric adopts permission-related mechanisms including policy, identity, channel, and private data collection (PDC). Use channels when entire transactions (and ledgers) must be kept confidential within a set of organizations that are members of the channel [11]. An example of this is the TradeLens system mentioned in Section 1.5. Channels are created for the different transport lines and for the parties who work together. Sensitive information including documents is distributed only to those nodes participating in a channel; this means that none of an ocean carrier's customer information will be distributed to other ocean carriers [13]. This is illustrated in Figure 3.6, where dots designate node participation in a channel.

The Fabric Gateway SDK allows applications to interact with a Fabric blockchain network. It provides a simple API to submit transactions to a ledger or query the contents of a ledger



	OCEAN CARRIER 1 CHANNEL	OCEAN CARRIER 2 CHANNEL	OCEAN CARRIER 3 CHANNEL	CHANNEL...
TRADELENS				
TRADELENS NODE	●	●	●	●
CARRIERS				
OCEAN CARRIER 1 NODE	●			
OCEAN CARRIER 2 NODE		●		
OCEAN CARRIER 3 NODE			●	
NODE...				●
OTHER PARTICIPANTS				
OTHER PARTICIPANT NODE	●		●	
OTHER PARTICIPANT NODE		●		
OTHER PARTICIPANT NODE		●	●	●

Figure 3.5: TradeLens channel separation [13].

with minimal code [4]. By content of a ledger, we mean the content of the blocks and assets that we can query and create. HLF provides the ability to modify assets using chaincode transactions. Assets are represented in Hyperledger Fabric as a collection of key-value pairs, with state changes recorded as transactions on a Channel ledger. Assets can be represented in binary and/or JSON form [10].

3.7 Private data collections

The TradeLens concept does not satisfy the requirements I have set because the decision factors would be visible to all other participants who provide information for the decision. This is also a problem when channels are created based on roles.

So I chose a solution for which I could find no instance, only mention as a potential solution. I worked with private data collections, which allow complete data separation and hiding in the decision-making process. Before I present the details of my solution, I will write briefly about PDCs.

Private data is stored in a different way from public data in the world state. Public data is stored in the form of (key, value, version) at all peers in the channel. Private data has two storage formats: the original (key, value, version) is stored at a subset of peers that are PDC members while the hashed (hashikey), hash(value), version) is stored at all peers [30]. The following diagram illustrates when the peer of the Central Credit Information

[4]Available here: `https://hyperledger.github.io/fabric-gateway-java/`

System submits an input value to its PDC. In the private state, the ledger contains the exact value, but the channel state contains only the hashed value. We can see in the Unauthorized Peer's ledger, there is only the hashed value of the input.

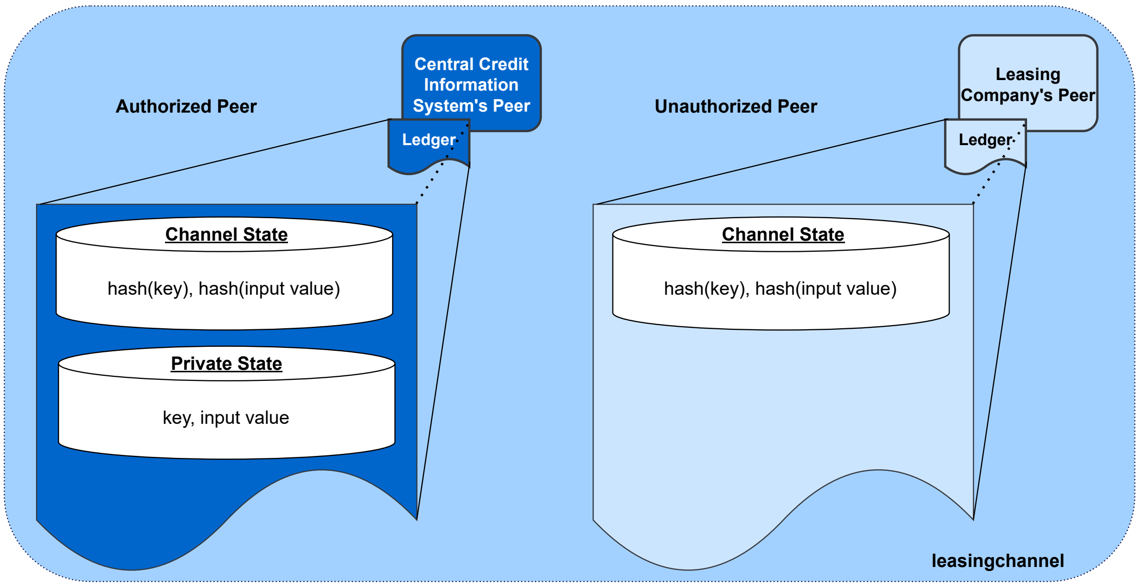


Figure 3.6: Ledger contents of an authorized and an unauthorized peer.

This allows a participant to store sensitive data in a hidden database from other participants. Unauthorized peers, on the other hand, can track changes to this data, via a version of the hashed value. This is why PDC is a great technology to use because it can protect the interests of both parties. One form of protection is hashing, which is a well-known method.

Definition 7. A **hash function** is a function that takes as input an arbitrarily long string of bits (or bytes) and produces a fixed-size result [18]. ▪

- **one-way function:** it can be computed but that cannot be inverted.
- **collision-free:** A collision is two different inputs i1 and i2 for which h(i1) = h(i2). This cannot occur in hashing.

3.7.1 Private data collection definition and transaction flow with private data

The definition is part of the chaincode definition. A collection definition can contain more than one collections. A collection definition has policy definition, which lists the organizations in the collection, this determines the authorized and unauthorized peers. It also contains properties that are used to control distribution of private data at the endorsement time. These values can make a difference in performance and can also depend on usage, so it is worth defining them by the use case.

When private data collections are referenced in chaincode, the transaction flow is slightly different in order to protect the confidentiality of the private data as transactions are proposed, endorsed, and committed to the ledger [11]. This mechanism can meet several requirements at the platform level. The transaction flow is illustrated in the following figure, where a transaction is submitted to a single-channel network containing 2 authorized, 2 unauthorized peers, and one orderer. The transaction is sent to an authorized peer.

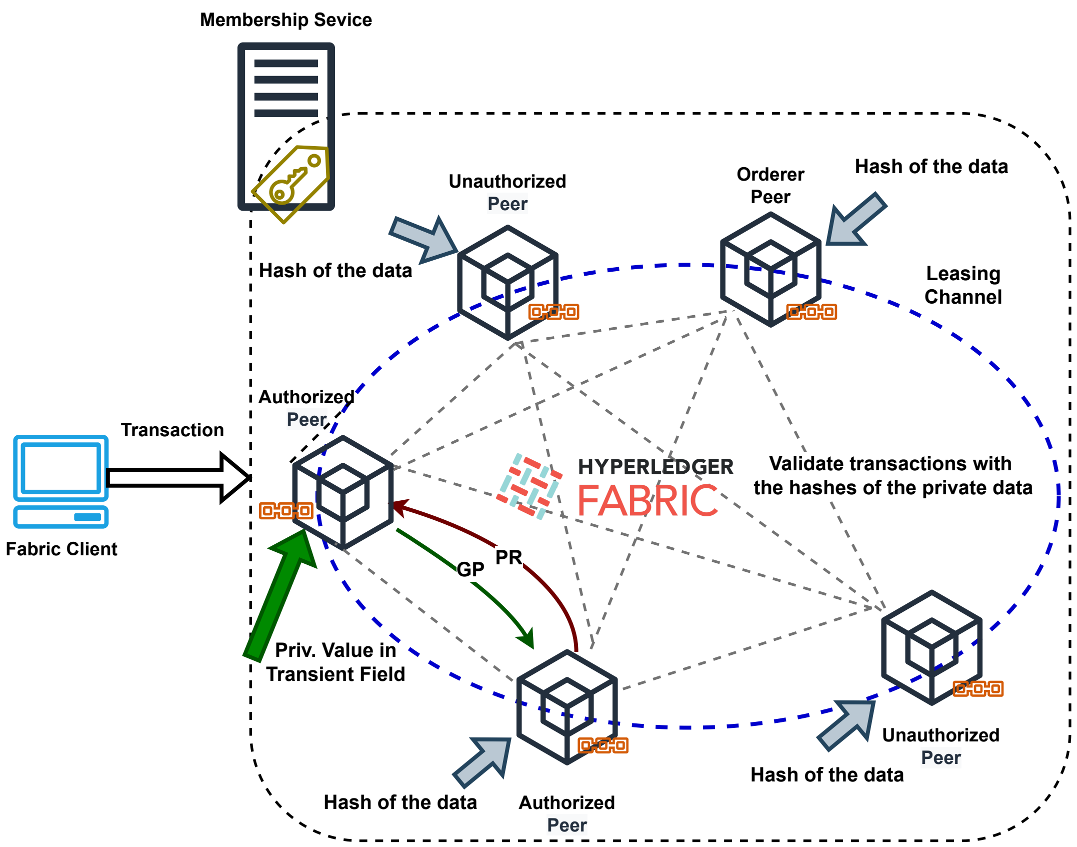


Figure 3.7: Transaction flow with private data.

Symbols and abbreviations used in the figure:

- **GP:** Gossip Protocol,
- **PR:** Proposal Response,
- **Grey arrows:** Shows that the hash of the data is distributed to each peer,
- **Blue dashed line:** Leasing channel,
- **Faint dotted grey lines between peers:** Shows that each peer on the channel validates the transaction with the hash of the private data in a consistent way.

The figure shows that the private value is received by one peer, and sent in the transient field. The collection definition defines the minimum number of authorized peers to forward the private value to, before the peer signs the endorsement and returns the proposal response back to the client. In this case, because there are multiple authorized peers, this is configured to 1, so the data is distributed using the gossip protocol.[5] The endorsing peers simulate the transaction and store the private data in a transient data store (temporary storage local to the peer). The orderer is not involved here and does not see the private data. However, as shown by the grey arrows, the block with the private data hash is distributed to all the peers. This serves as evidence to all peers to private data is stored. In this way, all peers on the channel can validate transactions with the hashes of the private data in a consistent way, without knowing the actual private data (faint dotted lines) [11]. The authorized peers will validate the private data against the hashes in the public block and commit the transaction and the block. Finally, the private data is deleted from the transient data store.

After the value is saved, in the collections definition, we can specify how long the private value should exist in the collection. Two other important attributes can be selected:

- memberOnlyRead: a value of true indicates that only the collection member organizations have read access to private data. Utilize a value of false if you would like to encode more granular access control within individual chaincode functions.
- memberOnlyWrite: a value of true indicates that only the collection member organizations have to write access to private data.

3.7.2 Benefits of private data collections

Setting the values of the properties mentioned previously allows a wide range of uses. When our data is stored in the PDC, it is ensured that it is inaccessible to others at both levels of block and chaincode. In contrast, we have the possibility to share and verify this data. There are several paradigms for sharing in the Hyperledger documentation. In my solution, I took advantage of verifiability and transparency. There are two options for verification, either the hash of the private data is known or the exact value is. In both cases, I offer the possibility to check the data. The version control allows us to track changes to private data in our system, even if we do not know its value. This also prevents the possibility of tampering.

[5]Available here: `https://hyperledger-fabric.readthedocs.io/en/latest/gossip.htmll`

Chapter 4

Blockchain-based decision management

Business processes that manage assets (e.g. transferring car/information/land titles) are a promising domain for applying blockchain technology: secure asset management (including tokens and cryptocurrency) is a major application area of blockchain [29]. I created a concept for decision-making where a ruleset is stored as an asset and it is executable.

4.1 Privacy-preserving decision-making

A set of rules stored as an asset is protected data, so its properties are not public, only the owner can see the whole table. Input data, the results of sub-calculations, and output data are also stored as assets. Access control is also being applied for these. The following figures show the data hiding used from the perspective of each role. To demonstrate masking, I chose an example where the input and output values can only take fixed (true, false) values. This illustrates that privacy-preserving decision-making also happens in such a case. If we look at a table with a box expression for example: "< 50". More visible and meaningful is what it means that unauthorized parties do not reach the exact value of the input. I would like to use this example to illustrate that even in such a case, our system can hide data and exclude the possibility of abuse. Of course, this is a simplified table for clarity. In the case of a table with more rules, the possibility of inference is even more excluded.

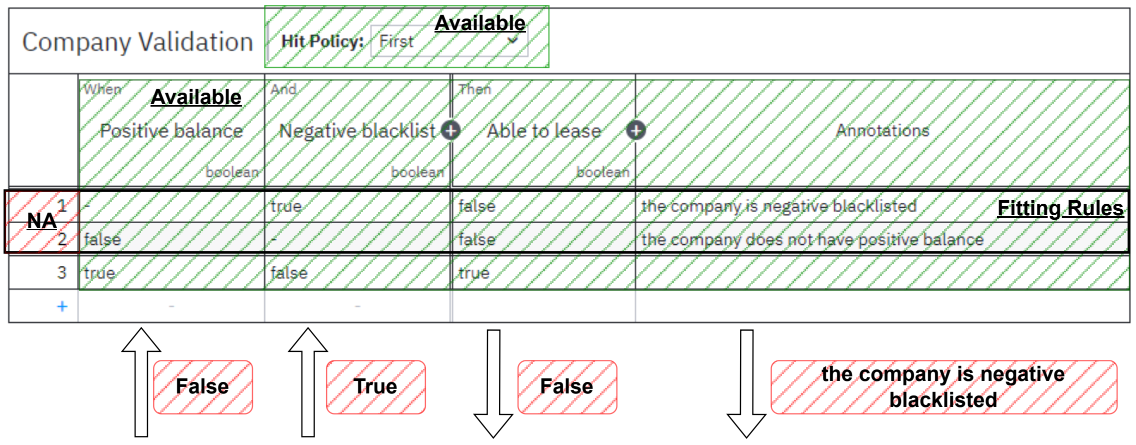
The marking system:

- Green area: accessible for the organizations.
- Red area: not accessible for the organizations.
- Blue area: not readable for the organization, but the contract can assess for evaluation.

Explanation of the figure:

- There are three rules, the hit policy is First.
- The given inputs are: Positive balance - false, Negative blacklist - true.
- For these inputs the fitting rules are: 1,2.
- Applied the hit policy, the first rule is the output.
- The output is: false, the annotation is: "the company is negative blacklisted".
- The inputs, outputs and ruleset belong to different organizations.
- In this example, each organization has only one role.

4.1.1 Ruleset owner


Company Validation — Hit Policy: First

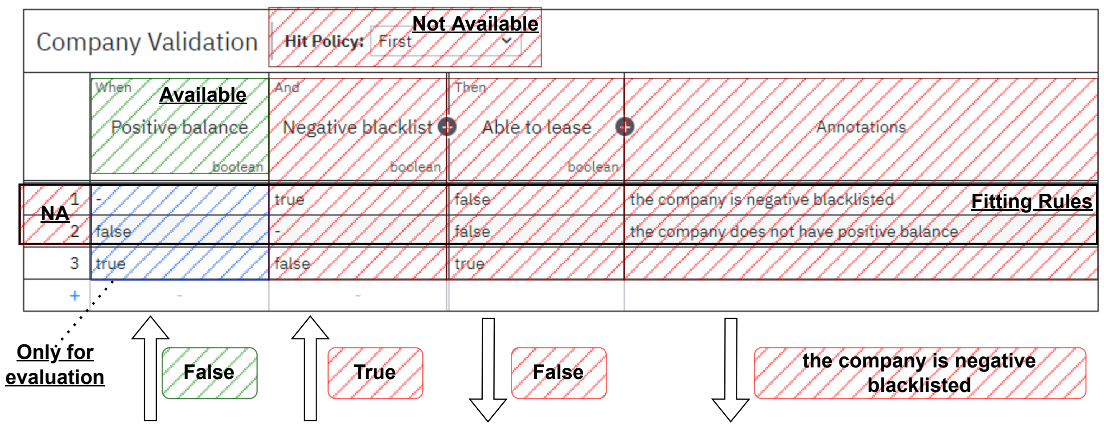
	When Positive balance (boolean)	And Negative blacklist (boolean)	Then Able to lease (boolean)	Annotations
1	-	true	false	the company is negative blacklisted
2	false	-	false	the company does not have positive balance
3	true	false	true	
+	-	-		

Figure 4.1: Information hiding from the ruleset owner.

The owner of the ruleset knows all the decision logic but nothing of the decision-making. This organization does not know the exact value of the inputs and outputs, and the calculation process is hidden from this user.

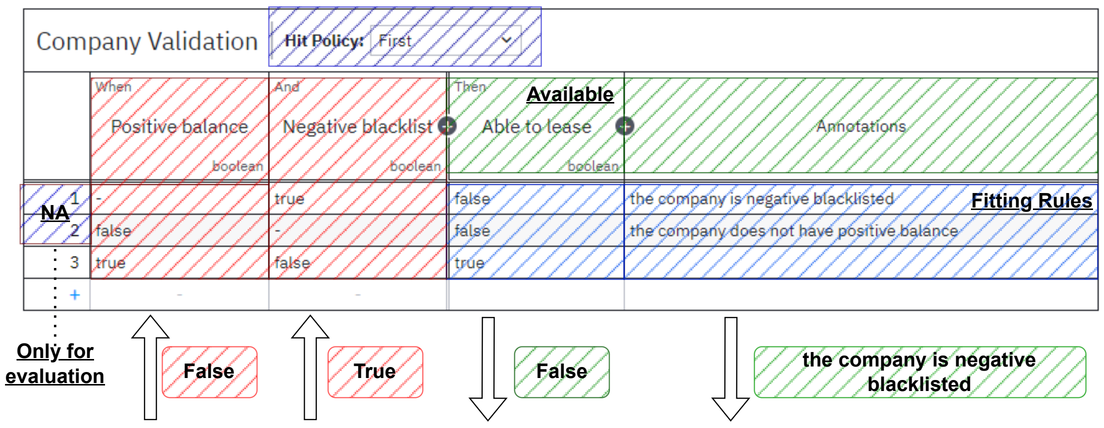
4.1.2 Input provider

In the figure below, you can see that the Positive balance input provider can read the name of the column. Of course, the input value it provides is also known by the input provider, but the values provided by the others are inaccessible. Similarly, the output is not accessible. The organization is unable to see the box values for its input or the hit policy. This prevents the possibility of choosing the value of the input on the basis of what is beneficial to them, instead of the real data. Once the own value is given by the organization, the result of the sub-calculation is for it: rule 1 and rule 2. This is knowable by the entity, but the calculations that follow are inaccessible.



	When Positive balance (boolean)	And Negative blacklist (boolean)	Then Able to lease (boolean)	Annotations
1	-	true	false	the company is negative blacklisted
2	false	-	false	the company does not have positive balance
3	true	false	true	
+	-	-		

Figure 4.2: Information hiding from an input provider organization.



	When Positive balance (boolean)	And Negative blacklist (boolean)	Then Able to lease (boolean)	Annotations
1	-	true	false	the company is negative blacklisted
2	false	-	false	the company does not have positive balance
3	true	false	true	
+	-	-		

Figure 4.3: Information hiding from an output receiver organization.

4.1.3 Output receiver

The output receiver knows the name of the output column, and the value of the output, and retrieves the value from the associated annotation column. The output receiver knows more data to calculate the final result, but cannot read them, only accesses it at computation time. These are the fields in blue, i.e. the hit policy, the result of the sub-calculations, and the possible values of the output. The results of the partial calculations are stored separately by the organizations providing the inputs. When calculating the output, we aggregate these and apply the hit policy. Thus, we can store the exact value of the output.

4.2 Architecture

The decision-making system is a Fabric network (decisionNetwork), where all parties involved in the decision are part of the blockchain. That means each company has a peer in the distributed system and each has private data collections (figure 4.5).[1] On the HLF network there are two contracts, and they have separated ledgers, these ledgers include

[1]Of course each peer has its world state, but I don't specify this in the diagram for simplicity.

world states and PDCs. I worked with a single channel (leasingChannel) network with an Orderer node. The single-channel solution has several advantages, all data will be tracked by the organizations, and the business logic (not the ruleset) will be visible to all, through the installed chaincode. Furthermore, we leave the possibility to create multiple channels, this can be used in a similar way as in the TradeLens example, for example, to separate the different groups. This can increase the transparency of transaction data.

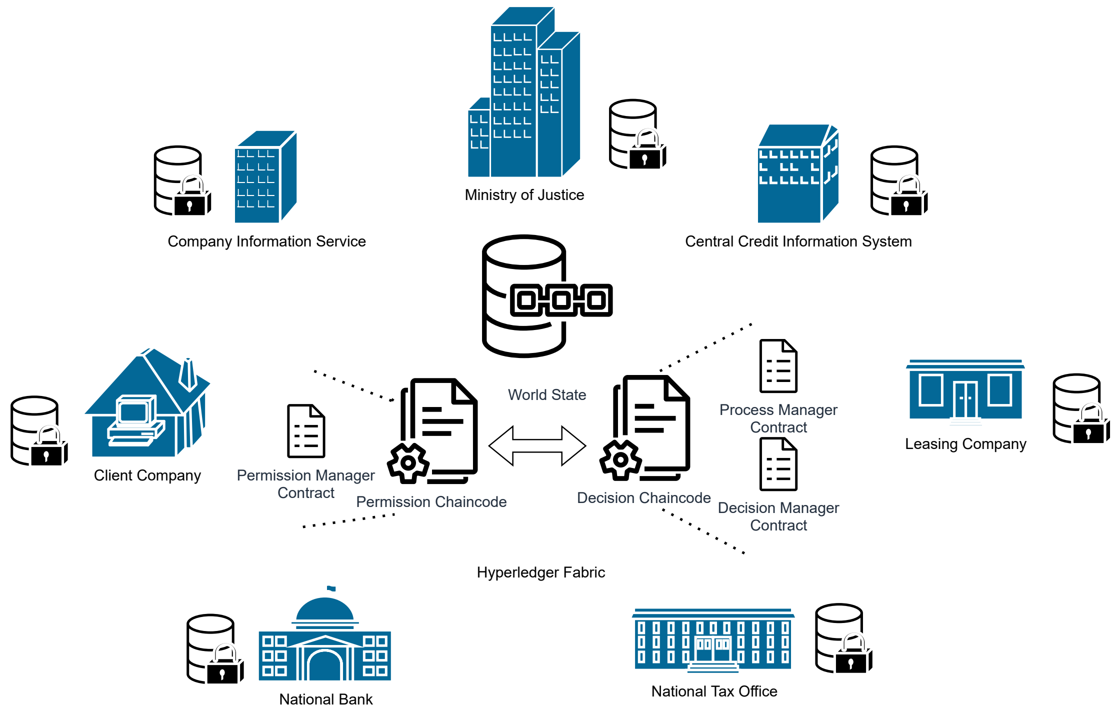


Figure 4.4: Organizations in the Validation decision with the network architecture.

Each organization has three private data collections managed by Decision Chaincode. They have different configurations, the most important of which is their readability and modifiability by unauthorized entities. The following figure illustrates this, where the arrows represent reading and writing rights.

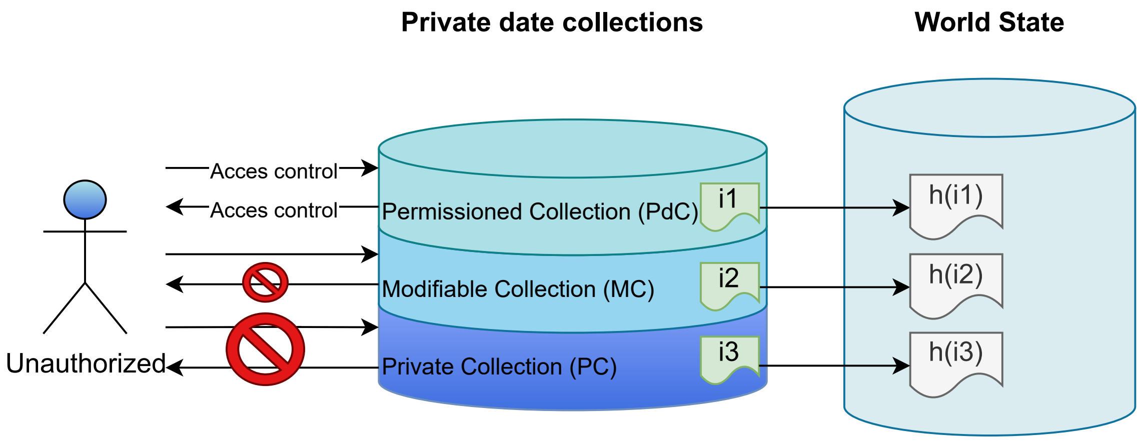


Figure 4.5: PDCs for an organization and the rights of an unauthorized entity.

It can be seen that access to Permissioned Collection data is not only controlled at the platform level, the addition to this collection is the Permission Manager Contract. Reading from or writing here, always involves a invoke of the Permission Chaincode.

4.2.1 Chaincode responsibilities

In my work, I implemented two chaincodes, the first is responsible for controlling access to private data, and the second has multiple responsibilities. The second one contains two separated contracts as we can see in Figure 4.4. Contracts and their responsibilities:

1. Permission Chaincode (PC):
    - Permission Manager Contract (PMC):
        - creating access rights for an object,
        - adding access rights for an object,
        - checking access rights for an object for an organization,
        - verifying rules from private data.
2. Decision Chaincode (DC):
    - Decision Manager Contract (DMC)
        - creating decision object from DMN,
        - adding data dependencies to the private data collections,
        - communicating with Permission Manager Contract to add access rights, and check them, when it is required,
        - verifying decision object from private data,
        - evaluating expressions,
        - enforcing hit policy.
    - Decision-making Process Manager Contract (DPMC)
        - start process,
        - communicate with Permission Manager Contract to add access rights, and check them, when it is required,
        - receiving input data of the decision,
        - querying the status of the process,
        - communicate with Decision Manager Contract to evaluate expressions,
        - querying the output of the decision.

Each chaincode has its world state and we can define private-data collections for them separately. Communication between chaincodes is available on Fabric in the way of cross-chaincode invocations. Within the Decision Chaincode, the Process Manager can access the Decision Manager Contract functions by contract instantiation.

4.3 Translating decisions to assets

In the next two sections, I will go through a scenario leading from the empty ledger to the point where a common decision has been made.

4.3.1 Create Table with a modeler tool

Before you can create a decision table on the blockchain, you need to create it with a suitable application (for example with Camunda Modeler). DMN is written in XML schema and is sent to the smart contract in this format. The properties for the table and columns are included as attributes in the XML. Rules, columns and associations are described as children.

4.3.2 Upload DMN to Ledger

The DMN file is sent as a string in the transient data field, this can be done using a client application. The upload is submitted to Decision Chaincode. Part of the Decision Manager Contract is an XmlParser class, which is responsible for processing the XML file. I used external libraries during the parsing process and saved the decision table parts in my own data model.

The libraries used are:

- Document Object Model (DOM) Level 3 Core:[2] this is a model for creating a *Document* object instance from the XML. Conceptually, *Document* is the root of the document tree and provides primary access to the document's data [3].
- Simple API for XML (SAX):[3] widely-used specification that describes how XML parsers can pass information efficiently from XML documents to software applications. The SAX parser uses the *InputSource* object to determine how to read XML input [5].
- javax.xml.parsers package:[4] provides classes allowing the processing of XML documents.

Once all data is received by reading the document tree, it becomes the properties of a *DecisionAsset* type instance. Which I designed for a reason other than the DMN metamodel. As you can see in the class diagram below, the table has been partitioned into input and output columns, instead of storing the set of rules in a row-by-row structure. The reason for this is that it makes it easier to manage the responsibilities and permissions associated with each column.

[2]Documentation: https://www.w3.org/TR/DOM-Level-3-Core/
[3]Documentation: http://www.saxproject.org/
[4]https://docs.oracle.com/javase/8/docs/api/index.html?javax/xml/parsers/package-summary.html

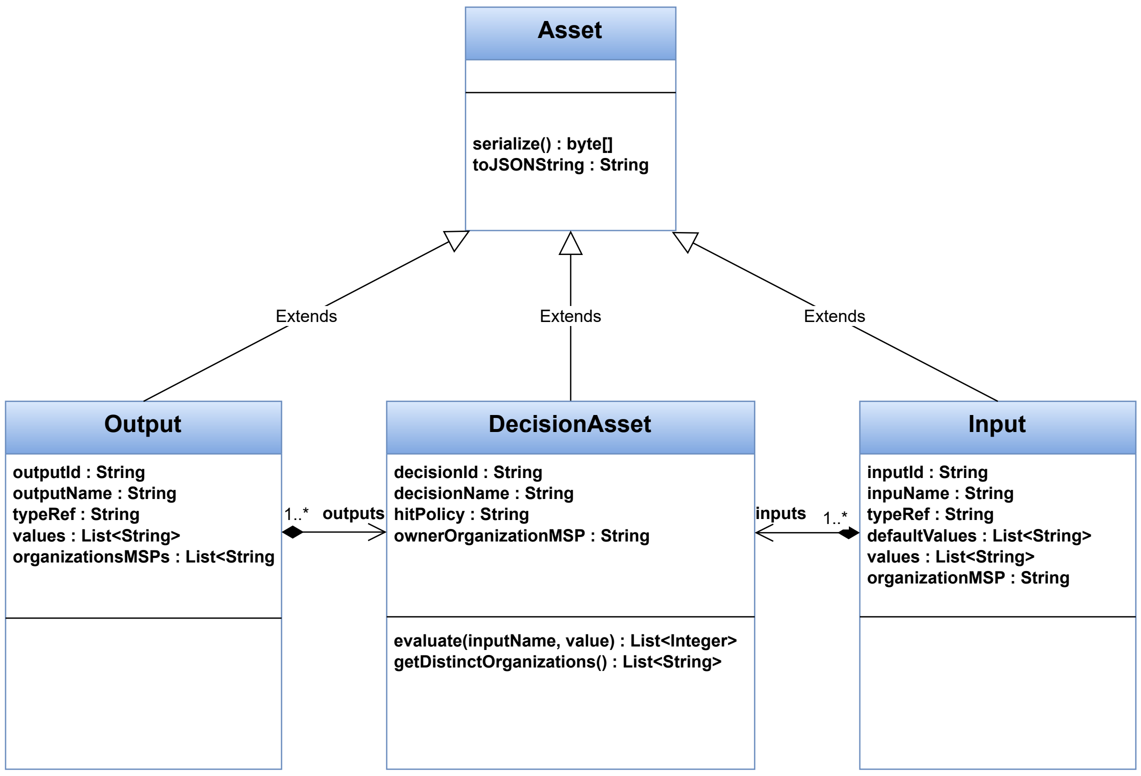


Figure 4.6: Class diagram of the data model.

The ruleset is stored in the permissioned collection of its uploader. Henceforth, a number of records will be saved to ledger automatically during the upload process. First, we save the permissions for the table using the Permission Manager Contract. These are:

- Owners of the rules system, they have the right to verify the table stored on the ledger.
- Input providers, they have the right to reach the input meta data and evaluate the column.
- Output receivers, they have the rights to reach the output meta data, calculating with the results of the sub-calculations, start decision-making processes and get outputs.
- Everyone who is part of the decision-making, they have the right to get the process status.

Then the contract save column meta data for organizations either provide input or receive output in a decision-making process. It contains the name and ID of the input/output and table, the input/output type and, if defined, the default values of the column. This record is stored in the modifiable collection. This prevents that when data needs to be provided, the user must first request the needed information from the ruleset owner. After that, a decision execution process can be started on the table.

4.4 Distributed evaluation

I will show the decision-making in table 1.3, where there are two input providers and one output receiver, the identity of these can be read from the DRD associated with the table (1.4).

4.4.1 Start process

The Leasing Company can start the decision-making process, it does this with a function in the Process Manager Contract. This does the following:

1. Queries its permissions in the decision-making.
2. Queries the organizations involved in the decision and adds a record of the start of the process to each organization's modifiable collection.
3. It creates a process object in its permissioned collection, which can be handled by other organizations, marking that the input has been provided. And the status can be read out to the participants.
4. Send an event with the process ID. This way, all organizations are informed that they have to provide data.

4.4.2 Enter input

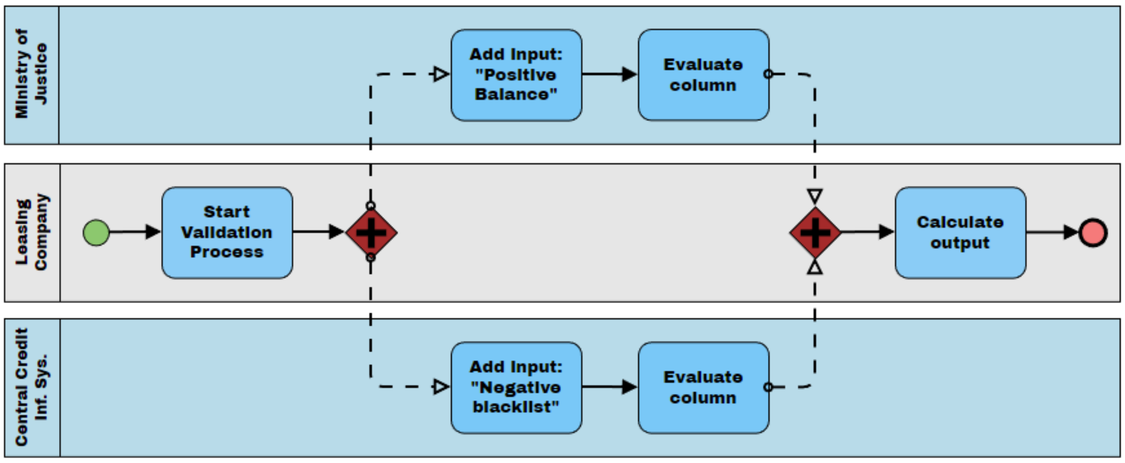


Figure 4.7: Decision-making process on Validation table, modeled using BPMN.

The following BPMN shows the evaluation process with the three organizations. The Leasing Company has started the process, and now the tasks in the two blue pools are next. The organization has been notified that it has to provide data, when it queries the data associated with the process ID, all the data is already in its own database. Thus, it can provide the input, which it saved in its own private data collection. In addition to saving, the sub-calculation is performed, which can be explained in the following steps:

1. Queries its permissions in the decision-making.
2. If the entity has access, it can read the box expressions belonging to the column for the calculation runtime.
3. Evaluates box expressions using the FEEL engine.
4. The contract saves the result in the organization's permissioned collection, the result means the rule IDs that fit its input.
5. In the process launcher's collection, the counter is decremented, which stores how many inputs are still missing.
6. If the value of the counter reaches 0, i.e. all input values have been given. An event is used to signal this.

4.4.3 Get output

After the Leasing Company receives the event, which indicates that both entities gave their input. This party's task is to aggregate the results and calculate the output. This can also be written up in steps:

1. Queries its permissions in the decision-making.
2. If the entity has access, it can read the results of the sub-calculations, the hit-policy and the values of the output column for calculation runtime.
3. The contract aggregate the results, and it applies the hit policy, thus, the decision-making is done.
4. Store the output of the decision-making.
5. The user reads out with an other function the output.

This brings us to the end of the decision-making process, the data and the results are still stored on the blockchain and can be retraced at any time. In case of conflict or disagreement, a conflict resolution is available.

4.5 Compliance with the requirements

Figure 4.7 shows the results of a test run I made. The figure is a representation of data protection and data separation in a specific abstraction view. Due to the complexity of the figure, I begin with its symbols and explanation and then demonstrate the fulfillment of the requirements set up in Chapter 2.

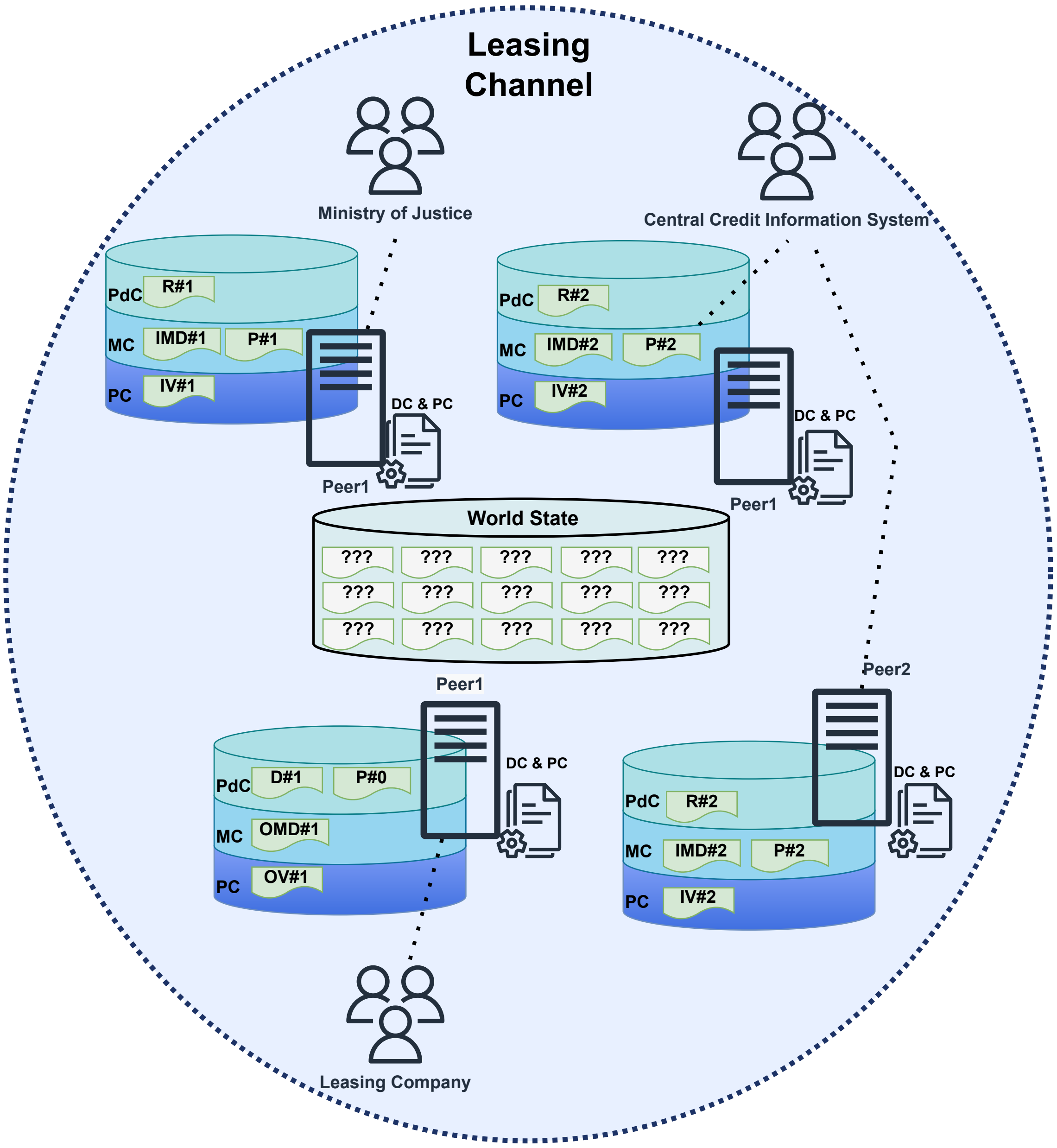


Figure 4.8: Sensitive data management in PDCs, modeled based on a test run.

4.5.1 Symbols and explanation

In the test I worked with the 1.3 table, the three organizations involved in the decision are part of the Leasing Channel. The Central Credit Information System as you can see has two peers and the other organizations have one-one. I deployed both chaincodes to all peers and all organizations have the three PDCs. For the sake of simplicity, the World State is shown only once, because its content is the same for all peers. The permissions stored by the Permission Chaincode are not shown in this figure, as it is stored separately from this data.

Symbol	Meaning
Blue surrounding ellipse	Leasing Channel
IV#1,IV#2	Records of Input Values
OV#1	Record of the Output Value
R#1,R#2	Records of Results of the sub-calculations
P#0, P#1, P#3	Records of the started Process
IMD#1, IMD#2	Records of Input Meta Data
OMD#1	Record of Output Meta Data
D#1	Record of the Decision table
PdC	Permissioned Collection
MC	Modifiable Collection
PC (in the cylinder)	Private Collection
DC	Decision Chaincode
PC (above the papers)	Permission Chaincode
???	Hashed value

Table 4.1: Symbols on the figure.

4.5.2 Requirements

To satisfy the requirements system, the requirements specified as a leaf must be fulfilled. In the following, I present methods to achieve these.

R.1.1.1 It is stored in D#1 PdC where we assign permissions to the available objects, thus not allowing editing of the table. If a user tries to write to a key that already contains data, the chaincode returns an error.

R.1.1.2 When providing input, it is not allowed by the contract to create/update a state with a key where data already exists. In addition, other organizations than the owner do not have access to this data.

R.1.1.3 Like D#1, R#1 (and R#2) cannot be edited by the above policy.

R.1.2.1 The asset used for this purpose is P#0, which can be edited along the permissions by the individual entities, and the current state of the decision process can be read.

R.1.2.2 By conflict resolution, we mean that the participants take the initiative to re-calculate the calculated values, in more critical cases to reveal and share the value of the given input values. All of these are provided by the contract, using methods that work with the data used in the first execution, thus eliminating the possibility of tampering.

R.1.2.3 If the modifiable data is changed, the public form stored in the world state is also versioned. In this way, we can track the modifications, and from the blocks, we can read the history, the modifier, and the block time of the transaction.

R.1.3.1 The function offered by the Decision Manager Contract provides a way to verify this, either hashed or with the original version. In this case, all the data stored in hashed format on the peers are also verified, thus increasing reliability.

R.1.3.2 As I wrote, we have the possibility to make a recalculation, and we also compare the hashed form of the input data with the value stored in the private storage.

R.1.4.1 Access to the table (D#1) is controlled at multiple levels, with PdC ensuring that the exact value is only stored by the owner, and PC controlling readability, with automatically generated permissions to determine which organizations can access which part of the table.

R.1.4.2 & R.1.4.3 These data are stored in a storage that only its owner is authorized to access, guaranteed by the PC belonging to the organization.

R.1.4.4 The transaction flow used by HLF for private data helps to fulfill at the platform level that sensitive data cannot be extracted from the communication channels, this is done by the transient field and the transient store.

R.2.1 By assigning the processing of the DMN to the chaincode, this requirement is satisfied, and there is no loss of efficiency and performance because the processing of the table is no longer required during executions.

R.2.2 Under different keys, the ledger can be used to store several tables. It is also possible to update an already installed table by adding it with a new ID and making the next decision-making on it. Decision-making that has already been started is performed on the original version.

R.2.3 The process object serves this purpose so that multiple evaluations can be triggered for a table.

R.2.4 With the MDD approach, these requirements are fulfilled, by uploading the table as a single DMN file, rather than as parts. From this file, for privacy-preserving decision-making, the permissions are automatically stored.

I did not include performance criteria in the non-functional requirements. The HLF can be configured at various levels, depending on the problem to improve performance. Improving performance and optimizing the configuration was not the focus of my research, its industrial compliance derives from data protection.

Chapter 5

Conclusion

In my work, I designed and implemented a method for privacy-aware decision-making. Decision-making is done by distributed evaluation, ensuring the confidentiality of all stakeholders.

I successfully created a DMN engine, which facilitates cooperative decision-making of mutually untrusted parties. For this purpose, I took advantage of the recent blockchain technologies and built an access control on a trusted schema. The decision-making method is model-driven and participants are not required to have any technical knowledge of smart contract programming. Decision-making is nevertheless transparent and guaranteed by a reliable platform, taking into account inconsistency, syntax or semantic errors, or gaps in the decision logic.

The architecture of my work can also be used in other fields of application. And my completed Business Rule Management System has a wide range of industrial and business applications. Because of the high level of access control, the model-driven execution, and the fact that its performance is highly predictable.

As a continuation of my work, I would like to work on embedding the DMN engine into the BPMN process. This requires a ready-made BPMN engine that works over a blockchain and, for example, bridges the two engines to make them work together. I also hope that my project can become an addition to the CBDC contract. However, the identity management used for CBDC is different from the way I manage it in decision-making. Therefore, future work could be a different approach to identity management.

List of Figures

Bibliography

[1] Business process model and notation (BPMN), version 2.0. page 532.

[2] Decision model and notation™ (DMN™) | object management group. URL https://www.omg.org/dmn/.

[3] Document object model (DOM) level 3 core specification. URL https://www.w3.org/TR/DOM-Level-3-Core/.

[4] What is role-based access control (RBAC)? examples, benefits, and more | UpGuard. URL https://www.upguard.com/blog/rbac.

[5] About SAX. URL http://www.saxproject.org/apidoc/overview-summary.html.

[6] DMN decision engine, . URL https://camunda.com/platform/decision-engine/.

[7] Camunda platform 8 docs | camunda platform 8 docs, . URL https://docs.camunda.io/.

[8] Hyperledger fabric – hyperledger foundation, . URL https://www.hyperledger.org/use/fabric.

[9] Hyperledger fabric – hyperledger foundation, . URL https://www.hyperledger.org/use/fabric.

[10] Hyperledger fabric model — hyperledger-fabricdocs main documentation, . URL https://hyperledger-fabric.readthedocs.io/en/latest/fabric_model.html#assets.

[11] Private data — hyperledger-fabricdocs main documentation. URL https://hyperledger-fabric.readthedocs.io/en/latest/private-data/private-data.html.

[12] Top rated business rules management vendors. URL https://www.peerspot.com/categories/business-rules-management.

[13] Solution architecture - TradeLens documentation. URL https://docs.tradelens.com/learn/solution_architecture/.

[14] S. H. Alsamhi and Brian Lee. Blockchain-empowered multi-robot collaboration to fight covid-19 and future pandemics. *IEEE Access*, 9:44173–44197, 2021. DOI: 10.1109/ACCESS.2020.3032450.

[15] Elli Androulaki, Artem Barger, Vita Bortnikov, et al. Hyperledger fabric: a distributed operating system for permissioned blockchains. In *Proceedings of the Thirteenth EuroSys Conference*, EuroSys '18, pages 1–15. Association for Computing Machinery. ISBN 978-1-4503-5584-1. DOI: 10.1145/3190508.3190538. URL https://doi.org/10.1145/3190508.3190538.

[16] Elisa Bertino. Rbac models — concepts and trends. *Computers Security*, 22(6):511–514, 2003. ISSN 0167-4048. DOI: https://doi.org/10.1016/S0167-4048(03)00609-6. URL https://www.sciencedirect.com/science/article/pii/S0167404803006096.

[17] Claudio Di Ciccio, Alessio Cecconi, Marlon Dumas, et al. Blockchain support for collaborative business processes. 42(3):182–190. ISSN 0170-6012, 1432-122X. DOI: 10.1007/s00287-019-01178-x. URL https://link.springer.com/10.1007/s00287-019-01178-x.

[18] Niels Ferguson, Bruce Schneier, and Tadayoshi Kohno. *Hash Functions*, chapter 5, pages 77–88. John Wiley Sons, Ltd, 2015. ISBN 9781118722367. DOI: https://doi.org/10.1002/9781118722367.ch5. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/9781118722367.ch5.

[19] Stephan Haarmann. *Executing DMN Decisions on the Blockchain*, pages 43–53. Springer International Publishing, Cham, 2021. ISBN 978-3-030-81409-0. DOI: 10.1007/978-3-030-81409-0_4. URL https://doi.org/10.1007/978-3-030-81409-0_4.

[20] Stephan Haarmann, Kimon Batoulis, Adriatik Nikaj, and Mathias Weske. Executing collaborative decisions confidentially on blockchains. In Claudio Di Ciccio et al., editors, *Business Process Management: Blockchain and Central and Eastern Europe Forum*, volume 361, pages 119–135. Springer International Publishing, . ISBN 978-3-030-30428-7 978-3-030-30429-4. DOI: 10.1007/978-3-030-30429-4_9. URL http://link.springer.com/10.1007/978-3-030-30429-4_9. Series Title: Lecture Notes in Business Information Processing.

[21] Stephan Haarmann, Kimon Batoulis, Adriatik Nikaj, and Mathias Weske. DMN decision execution on the ethereum blockchain. In John Krogstie and Hajo A. Reijers, editors, *Advanced Information Systems Engineering*, volume 10816, pages 327–341. Springer International Publishing, . ISBN 978-3-319-91562-3 978-3-319-91563-0. DOI: 10.1007/978-3-319-91563-0_20. URL http://link.springer.com/10.1007/978-3-319-91563-0_20. Series Title: Lecture Notes in Computer Science.

[22] Faruk Hasić, Carl Corea, Jonas Blatt, et al. A tool for the verification of decision model and notation (dmn) models. In *Research Challenges in Information Science*, pages 536–542, Cham, 2020. Springer International Publishing. ISBN 978-3-030-50316-1.

[23] Jan Mendling, Ingo Weber, Wil Van Der Aalst, et al. Blockchains for business process management - challenges and opportunities. 9(1):1–16. ISSN 2158-656X, 2158-6578. DOI: 10.1145/3183367. URL https://dl.acm.org/doi/10.1145/3183367.

[24] Yanqing Peng, Min Du, Feifei Li, Raymond Cheng, and Dawn Song. Falcondb: Blockchain-based collaborative database. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, SIGMOD '20, page 637–652, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450367356. DOI: 10.1145/3318464.3380594. URL https://doi.org/10.1145/3318464.3380594.

[25] Pierangela Samarati and Sabrina Capitani de Vimercati. Access control: Policies, models, and mechanisms. In Riccardo Focardi and Roberto Gorrieri, editors, *Foundations of Security Analysis and Design*, Lecture Notes in Computer Science, pages 137–196. Springer. ISBN 978-3-540-45608-7. DOI: 10.1007/3-540-45608-2_3.

[26] Pierangela Samarati and Sabrina Capitani de Vimercati. Access control: Policies, models, and mechanisms. In Riccardo Focardi and Roberto Gorrieri, editors, *Foundations of Security Analysis and Design*, pages 137–196, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg. ISBN 978-3-540-45608-7.

[27] B. Selic. The pragmatics of model-driven development. *IEEE Software*, 20(5):19–25, 2003. DOI: `10.1109/MS.2003.1231146`.

[28] William Tolone, Gail-Joon Ahn, Tanusree Pai, and Seng-Phil Hong. Access control in collaborative systems. *ACM Comput. Surv.*, 37(1):29–41, mar 2005. ISSN 0360-0300. DOI: `10.1145/1057977.1057979`. URL `https://doi.org/10.1145/1057977.1057979`.

[29] An Binh Tran, Qinghua Lu, and Ingo Weber. Lorikeet: A model-driven engineering tool for blockchain-based business process execution and asset management. page 5.

[30] Shan Wang, Ming Yang, Yue Zhang, et al. On private data collection of hyperledger fabric. In *2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS)*, pages 819–829. DOI: `10.1109/ICDCS51616.2021.00083`. ISSN: 2575-8411.

[31] Gavin Wood. Ethereum: A secure decentralised generalised transaction ledger. URL `https://blossom.informatik.uni-rostock.de/28/`. Publisher: Etherum.

[32] Dylan Yaga, Peter Mell, Nik Roby, and Karen Scarfone. Blockchain technology overview. URL `https://nvlpubs.nist.gov/nistpubs/ir/2018/NIST.IR.8202.pdf`.

[33] Svein Ølnes, Jolien Ubacht, and Marijn Janssen. Blockchain in government: Benefits and implications of distributed ledger technology for information sharing. *Government Information Quarterly*, 34(3):355–364, 2017. ISSN 0740-624X. DOI: `https://doi.org/10.1016/j.giq.2017.09.007`. URL `https://www.sciencedirect.com/science/article/pii/S0740624X17303155`.