



Article

Blockchain-Driven Generalization of Policy Management for Multiproduct Insurance Companies

Abraham Romero ^{1,*} and Roberto Hernandez ^{2,*}¹ Industrial Engineering Doctoral Program, Universidad Nacional de Educación a Distancia (UNED), 28040 Madrid, Spain² Department of Communication and Control Systems, ETSI Universidad Nacional de Educación a Distancia (UNED), 28040 Madrid, Spain

* Correspondence: aromero471@alumno.uned.es (A.R.); roberto@scc.uned.es (R.H.)

Abstract: This article presents a Blockchain-based solution for the management of multipolicies in insurance companies, introducing a standardized policy model to facilitate streamlined operations and enhance collaboration between entities. The model ensures uniform policy management, providing scalability and flexibility to adapt to new market demands. The solution leverages Merkle trees for secure data management, with each policy represented by an independent Merkle tree, enabling updates and additions without altering existing policies. The architecture, implemented on a private Ethereum network using Hyperledger Besu and Tessera, ensures secure and transparent transactions, robust dispute resolution, and fraud prevention mechanisms. The validation phase demonstrated the model's efficiency in reducing data redundancy and ensuring the consistency and integrity of policy information. Additionally, the system's technical management has been simplified, operational redundancies have been eliminated, and privacy is enhanced.

Keywords: blockchain; multipolicy management; Merkle tree; insurance; smart contracts; policy standardization



Citation: Romero, A.; Hernandez, R. Blockchain-Driven Generalization of Policy Management for Multiproduct Insurance Companies. *Future Internet* **2024**, *16*, 356. <https://doi.org/10.3390/fi16100356>

Academic Editor: Petros Patias

Received: 19 August 2024

Revised: 27 September 2024

Accepted: 28 September 2024

Published: 30 September 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

The insurance market is one of the most important economic sectors globally, directly influencing financial stability and economic development worldwide. In 2023, the global insurance sector achieved revenue exceeding 7 trillion USD, with substantial growth projections fueled by emerging markets and the continuous increase in demand for insurance-related products. The insurance market is not only about protecting individuals or property; it also plays a crucial role in long-term capital investment, fostering the creation of global partnerships and projects.

In the insurance market, it is possible to find companies of various types. Some companies specialize in a single type of insurance, such as auto, home, life, or health insurance, offering specialized services in one of these areas. However, it is increasingly common for insurers, even the most traditional ones, to offer a broad portfolio of different types of insurance. Today, thanks to the rise and evolution of digital markets, companies and organizations that are not traditionally associated with the insurance industry (such as telecommunications, energy, or retail companies) are offering different types of insurance to their customers. To achieve this, these companies have entered into agreements and collaboration contracts with various insurers to provide these services. For example, the U.S. company Apple offers device coverage services through AppleCare, in partnership with the multinational AIG. Similarly, the globally recognized e-commerce company Amazon provides the option to purchase protection insurance for certain products through Amazon Protect, an alliance with the British insurer London General Insurance Company Limited.

Whether dealing with multi-insurance companies or collaboration agreements between different entities aimed at offering multiple types of insurance, one of the primary

assets they handle is customer data. Since insurance policies are involved, sensitive and private data such as banking details, medical history, beneficiary information, traffic violations, wills, and more are stored, transferred, and managed. All this information must be rigorously safeguarded, ensuring its privacy and integrity.

The advancement of technology has led to an increasingly sophisticated process of digitalization within the insurance industry. As shown by Eling and Lehmann [1], digitalization is transforming customer interactions, business processes, and the creation of new products, while also presenting challenges related to information asymmetry and system connectivity. Moreover, by leveraging the vast amount of data from various sources (IoT, web, leads, etc.), a significant digital transformation has been achieved across many systems involved in the insurance process, including marketing, sales, and customer support. This is highlighted in Albrecher et al. analysis [2], which concludes that the digitalization of these systems enables better product personalization, process optimization, and greater operational efficiency through the use of advanced data analysis techniques and digital models, achieving a level of sophistication never seen before.

If we analyze most of the IT solutions that underpin the current insurance business model (see [3] for details), a series of processes, systems, and software applications can quickly be identified that require complex relationships between different participants. This leads to a high number of redundancies, functional dependencies, and technical complexity in both the management and maintenance of these systems. Such complexity introduces risks, including the potential for fraud and cyberattacks targeting the availability, integrity, or confidentiality of information. Examples of these attacks include those suffered in 2021 by CNA Financial Corporation, one of the leading U.S. companies, where cybercriminals repeatedly accessed their systems, extracting information and finally infecting their systems with ransomware. Similarly, in 2023, Aviva, one of the most prominent British insurers, experienced a security breach that compromised the personal information of its customers.

Given the importance of the insurance industry in the social and economic spheres, it has frequently been the focus of study and improvement through the application of various technologies over the past few decades. A survey of these applied technologies can be found in [4], with one of the most recent and notable being the application of blockchain technology.

The overall objective of this work is to provide solutions to the most frequent and significant issues encountered in the current insurance market management. These problems include:

- Ensuring document integrity to avoid litigation and manipulation in processes, a particularly serious issue in claims and legal proceedings. For example, there are cases where the insurance company holds a different document than that provided by an external collaborator, as occurred in *Corbin & King Ltd. vs AXA Insurance UK Plc* in 2022, where discrepancies arose between the documents submitted by the parties regarding coverage under the NDDA (Denial of Access) clause in their policies. Such cases typically end with an expert review in court to determine liability in a claim. To address this challenge, blockchain technology has positioned itself as a suitable tool due to its ability to ensure data integrity through immutable and auditable records, as demonstrated in existing literature.
- Fraud management occurs when documents are manipulated, discrepancies arise between data in different systems, or there is a lack of transparency in third-party operations. Blockchain addresses these issues by immutably recording each transaction. For example, a common type of insurance fraud involves altering policy data, filing a claim by exploiting discrepancies in cancellation or expiration dates, or a provider inflating the amount on an invoice. With blockchain, such alterations would be impossible without detecting a discrepancy in the blockchain, thanks to automated verification mechanisms provided by smart contracts.
- System and data security. Traditional systems can be vulnerable to attacks that compromise the integrity and privacy of information. In contrast, blockchain distributes data

across multiple nodes, meaning that attacking a single node would not compromise the entire system. This provides greater resilience against attacks and prevents data theft or manipulation. Furthermore, in blockchain—and with the proposed solution using Merkle trees to model the standardized policy—the sensitive data are protected through hashing, and privacy is ensured by encryption mechanisms.

- The importance of data in the insurance sector is critical. While a company may operate with redundancies in its processes, it can face serious issues if data are lost or manipulated. For instance, if an external provider holds a set of data that does not match the insurer's records, it can lead to conflicts resulting in costly litigation and reputational damage. With blockchain, such problems are avoided, as all participants have access to the same verified information with no possibility of manipulation.

Once the general issues of the sector have been reviewed, it is also necessary to identify the specific technical challenges that are frequently encountered in the management of policies and insurance processes. Among these, the following stand out:

- **Data and process redundancy:** Traditional systems operate with multiple workflows containing redundant data, which leads to high maintenance costs and significant technical complexity, particularly when scaling the solution. For example, customer information is often replicated across all external systems involved. Additionally, there are fragmented processes across systems or departments performing the same function, such as data input or transmission. Blockchain provides a comprehensive and decentralized structure that eliminates data and process duplication by allowing all parties to access a single source of information. In this model, all information is inserted, queried, and modified in the distributed ledger, ensuring integrity, immutability, and ease of verification.
- **Difficulty in scaling operational processes:** The need to incorporate new products into the insurer's portfolio, especially in environments with multiproduct policies, presents significant challenges. For example, if a company manages life, automobile, and health insurance and wants to address the current issue with electric scooters by adding a specific policy for this modality, it may require substantial changes affecting both the system's core and third-party applications and systems. Blockchain, combined with the proposed policy standardization, enables efficient scaling through the use of Merkle trees as an anchoring mechanism and smart contracts that automate policy management. This approach ensures that the addition of new products does not compromise the system's integrity or efficiency, and that information already stored remains intact under any circumstances. Furthermore, this model contributes to reducing operational and maintenance costs by lowering the technical complexity associated with product expansion and data management.
- **Data inconsistencies between systems:** Insurers and third parties often manage different versions of the same information, leading to inconsistencies. For instance, it is common to find discrepancies in data such as dates, premium amounts, or customer details across various participants in the process. Blockchain ensures that all parties have access to the same synchronized and verifiable data, preventing errors or discrepancies that could lead to legal or financial consequences.
- **Operational efficiency:** Stemming from the issues mentioned above, traditional systems can experience efficiency problems, leading to higher maintenance costs, operational overload, delays, incidents, and more. These problems often affect customer service, resulting in complaints and claims with potential reputational impact. With the proposed solution, we aim to create a more efficient system that optimizes operational times. The inclusion of Merkle trees and the use of smart contracts provide improvements in data verification and retrieval, which is the most frequent operation in such systems.

For this purpose, a prototype has been built as a proof of concept based on a private Ethereum network using Hyperledger Besu for transaction management and Tessera to ensure privacy in sensitive transactions. This framework offers a decentralized, scalable

solution that reduces redundancies, enhances fraud prevention, and ensures consistency across all entities managing multiproduct insurance portfolios. To summarize, we make the following contributions:

- First, standardization of the policy model is carried out, allowing the same insurance company to manage multiple types of insurance simultaneously.
- Following this, a blockchain architecture is proposed, analyzing its advantages and determining the characteristics it must possess to effectively handle these multipolicies.
- A Merkle tree-based anchoring protocol will be presented, enabling the integration of the standardized model into the blockchain. Additionally, it will facilitate the precise verification of changes made to the data.
- Finally, a prototype has been developed to enable a proof of concept for the creation of multiproduct policies in a simulated environment. This prototype aims to demonstrate the ease and efficiency of generating different types of insurance products quickly and seamlessly.

The article is organized as follows: Section 2 describes existing studies and applications of blockchain in areas related to the insurance sector. Section 3 outlines the problem statement, focusing on the current policy management model in the insurance market. Section 4 provides an analysis of the proposed solution, presenting a standardized policy model and a technological alternative based on blockchain and Merkle trees. Section 5 demonstrates how the solution has been implemented, detailing each component of the proposed architecture. Finally, Section 6 concludes.

2. Related Work

The application of blockchain technology to the insurance sector has gained significant attention in recent years, with numerous efforts exploring how blockchain can be used to address security, transparency, and efficiency challenges.

In the current insurance market, it is important to recognize that traditional technological systems are generally used due to their long-standing existence and the significant resources already in place. However, there have been some noteworthy proposals related to blockchain. For instance, the challenges associated with adopting this technology in insurance applications have been theoretically addressed with the aim of establishing a decision-making framework to facilitate its adoption [5]. Additionally, specific frameworks have been proposed for the work needed to apply blockchain in insurance processes, based on a comprehensive study of the state of the art in this technology, the insurance market, and the requirements for fraud prevention and claims management [6]. Similarly, frameworks and applications based on smart contracts have also been proposed [7]. The goal of these studies is to explore the potential economic and social transformation that blockchain could bring to the insurance industry. Raikwar et al. [8] propose a blockchain framework for insurance processes, emphasizing transparency and transaction auditability. Their work demonstrates how blockchain can enhance the management of insurance policies through an immutable ledger, although it does not focus on a multiproduct model.

Various theoretical approaches have been made; for example, Amponsah et al. [9] provide an exploratory analysis of the prospects and threats of blockchain in the insurance sector, highlighting both the opportunities for enhancing data integrity and privacy and the challenges posed by regulatory issues and technological adoption. This work aligns with our approach to leveraging blockchain for policy management, providing additional insight into the potential barriers and drivers of blockchain implementation in insurance.

Practical approaches can also be found. Zhou et al. [10] introduce MISore, a blockchain-based medical insurance storage system designed to securely manage and store payment information related to health insurance. Their work explores the use of blockchain for efficient data storage and accessibility, focusing on healthcare applications. This is particularly relevant as it demonstrates the potential of blockchain in securely handling sensitive insurance data, aligning with the security and integrity objectives of our proposed solution in the multiproduct insurance domain using an approach implemented on the public Ethereum

network [11]. Loukil et al. [12] introduce Ciosy, a blockchain-based collaborative insurance system that facilitates secure data exchange between insurers and customers. Their work highlights blockchain's potential to reduce administrative costs and enhance data privacy, though it primarily focuses on customer-facing processes rather than internal multiproduct management. Finally, Dutta et al. [13] investigate the use of blockchain smart contracts for parametric insurance in transport and logistics. This work is valuable in understanding how blockchain can automate specific insurance processes.

Some of them, although not directly related to the insurance market, propose interesting approaches that are easily applicable to our solution.

Yu et al. [14] propose an efficient and privacy-preserving medical research support platform based on blockchain, designed to facilitate secure data exchange in response to the COVID-19 pandemic. Their work emphasizes the importance of privacy in medical research data, using blockchain to ensure data integrity and confidentiality. This approach highlights the broader applicability of blockchain in securely managing sensitive data, which is directly relevant to our exploration of blockchain for policy management in the insurance sector.

Mani et al. [15] introduce Hyperledger Healthchain, a patient-centric blockchain system utilizing IPFS for the secure storage of health records. Their work demonstrates the potential of combining blockchain with decentralized storage to ensure data privacy and accessibility in healthcare.

Bianchini et al. [16] explore blockchain integration in supply chains, emphasizing a resource-oriented approach. Although this research is centered on supply chains, the principles they apply regarding interdependency management and blockchain integration are applicable to multiproduct insurance scenarios, as addressed in this paper.

Finally, Yadav et al. [17] present a privacy-preserving blockchain system for vehicle accident and insurance registration. Their work is highly relevant for understanding how blockchain can handle sensitive data securely.

A very important aspect when we talk about insurance is fraud management. Fraud is one of the main challenges of the insurance market. And in this line we find work such as Nath [18] presents a blockchain-based data exchange platform aimed at combating fraud in the insurance industry. This platform focuses on improving data transparency and security through the decentralized nature of blockchain, allowing for more efficient fraud detection and prevention. Also, Dhieb et al. [19] propose an AI-driven architecture designed to automate insurance systems, focusing on fraud detection and risk measurement. Their solution leverages AI algorithms to enhance the security and efficiency of insurance operations, providing real-time analysis and risk assessment.

In contrast to these works, our solution focuses on providing a flexible and scalable multiproduct policy management system. The standardization of policies is the key aspect that allows insurers to manage various products efficiently, offering the flexibility and scalability required to adapt to the increasing complexity of the sector. Additionally, the use of Merkle trees as an anchoring mechanism ensures efficiency in data verification. All of this has been integrated into a private Ethereum network, ensuring privacy and access control to the information. This combination fills a gap in the literature by merging standardization with blockchain technology for multiproduct management.

3. Problem Statement

3.1. The Insurance Market: Policy Management

When we talk about the insurance market, the most important element we deal with is the insurance policy.

A policy is a contract between the policyholder and the insurance company. The policyholder may or may not be the insured party under the policy. This distinction introduces the concept of the insured, who is the person or object covered by the policy. Additionally, there is the figure of the beneficiary, who is the individual or entity that will receive the compensation, if applicable.

In general, a policy can be composed of several key elements: the policyholder's information, the insured and beneficiary's details, and information about the object to be insured, such as a house, a car, or the insured person's life. Additionally, the policy includes a set of coverages and guarantees. These are essential components that define what is being insured and under what types of incidents or risks. For example, when insuring a home, we refer to content or structure coverage. In the case of automobiles, we have coverage options like glass breakage, theft, etc. For health insurance, coverage may include disability leave, surgery, and more.

Another key concept is the premium. The premium refers to the annual or monthly cost of keeping the insurance contract in force. In the event of a claim, the insurance company will be responsible for making the necessary arrangements or providing compensation as outlined in the policy's coverages and guarantees. The price of the premium is calculated by taking into account various factors such as the policyholder's age, the age or condition of the insured asset, past claims, the place of residence, and other relevant criteria.

Finally, it is important to review the dates of contract initiation, validity, expiration, and any modifications. Proper management of this information is crucial to ensuring the consistency of operations throughout the entire life cycle of a policy, particularly in the handling of claims.

The most common way insurance companies manage policies from the moment of contract initiation is through batch processing. Each day, typically overnight, records from all possible company sources—such as online applications, branch offices, insurance brokers, etc.—are uploaded. The upload is usually performed at night to avoid potential impacts on operational systems. Finally, through automated processes, all policies are processed and validated before being finalized. The goal of this step is to confirm that the reported data aligns with the company's criteria.

The implementation of the policy concept in these systems consists of three main elements: data acquisition, policy storage, and policy management. Data acquisition occurs in real-time through web forms at the point of contract initiation. Once the data are validated and the policy is approved, it is stored in a database system, typically a relational database. This allows for the creation of the data model containing all the information within the policy. Finally, additional systems are required for policy management, enabling the sharing and modification of all information. It is common to use APIs, web services, and batch processing systems to perform these tasks.

The rapid evolution of the insurance market and the growing commercial relationships between different companies have highlighted the necessity to cover a broader spectrum of business. This has led companies that were traditionally focused on a specific type of insurance to expand their operations towards a multipolicy model.

Today, companies no longer offer a single type of policy, such as auto, home, or health insurance, but can provide multiple types of products simultaneously. As a result, there is now a necessity for a management process adapted to this new data model, where a single customer may hold different types of policies.

From a technological perspective, so-called Policy Administration Systems (PASs) are commonly used. These systems enable the configuration and management of various insurance products from a centralized platform. Among their key components are:

- Creation of templates for each type of insurance (auto, life, health, home), including their own coverages, business conditions, and economic data.
- Customized parameterization for each policy type, such as coverage limits, deductibles, and premiums.
- Rule engine specific to each type of insurance, enabling tailored evaluation processes.
- Centralized policy management using specific modules for each type of insurance.

Here is an example to understand how these systems work. If an insurance company offers auto, life, and home insurance, they would use a PAS as follows:

- Auto Insurance: A template with coverages such as collision, liability, and roadside assistance.

- Life Insurance: A template with options for term life, whole life, and universal life insurance.
- Home Insurance: A template with coverages for fire, theft, and natural disasters.

The complexity of these systems lies in data management, which requires efficient and well-designed modeling. Typically, a relational database model is used, with tables supporting each of the different types of insurance. Additionally, when dealing with multiple products simultaneously, all systems that are part of this type of architecture must be equipped to support this multiproduct model.

Some of the main problems that may arise are:

- Duplication of Processes: In the management of insurance business operations, it is common to implement similar processes across multiple systems. This can lead to operational overload, increased implementation and maintenance costs, and reduced efficiency due to redundancy and the necessity for synchronization between systems.
- Duplication of Data: When both the insurance company and other involved parties store information about customers, policies, or claims, duplications can occur across different systems.
- Inconsistencies: If customer or policy data are updated in one system but not in another, inconsistencies can arise. These can also result from errors during data integration or transmission. For example, different versions of a policy might exist in various systems due to an update error.
- Discrepancies in Risk Assessment: When the insurer and other entities operate independently, they may use different criteria or models for risk assessment. These differences are reflected in their technological architectures, leading to inconsistencies even when agreements are made to conduct operations or transfers. Examples include discrepancies in premium assignments, policy approvals, and claims resolution.
- Inconsistent Appraisal Reports: Appraisal results can vary if the methodologies or data used by external entities differ from those of the insurer.
- Fraud Risk: This is one of the primary challenges faced by insurance companies. The risk of fraud can arise in various ways due to the involvement of multiple actors and integration points. Some of the most common examples include:
 - Customer Fraud: For example, registering duplicate claims in an attempt to receive double compensation.
 - Data Manipulation: An agent or external entity might alter customer or insured asset information to artificially reduce the insurance premium or to approve policies that would normally be rejected. Another method involves inflating the value of damages to obtain a larger compensation.
 - Collusion: Insurance company employees may collude with external personnel to approve fraudulent claims or manipulate risk assessments.

These examples highlight the challenges of a centralized data model that requires numerous interactions between different entities. This is why our initial approach involves the standardization of the policy model, making it possible to manage multiple products under a unified data model.

3.2. Analysis of the Traditional Architecture of an Insurance Company

Based on a comprehensive insurance management system, we conducted an analysis aimed at identifying areas for improvement. The goal is to generalize the information in a way that it can be applied to any type of company and insurance branch, whether it be life, health, auto, home, etc.

We conducted the analysis in two distinct phases. In the first phase, we reviewed the complete operational flow that a customer or policyholder can engage in from the moment they purchase a policy, regardless of the insurance branch or the nature of the insurance company.

To achieve this, we adopted a use-case-based methodology [20], from which the following use cases were defined.

Figure 1 illustrates these use cases. In general, the customer has five distinct groups of actions:

- **Policy Management:** This encompasses the initiation of the commercial relationship between the insured and the insurance company. It includes operations such as policy issuance and cancellation, management of policy guarantees and coverages, and premium payments.
- **Claim Filing:** This is one of the most common operations in the insurance market. When the insured experiences a loss, regardless of the type of insurance purchased, a claims process is initiated. For example, if it is a homeowner's insurance policy and a water pipe has burst.
- **Damage Assessment:** Sometimes, after a claim is filed, the insurance company or the claims processing firm may require appraisal services to assess the declared damages. This serves as a first barrier against fraud.
- **Repair or Compensation:** When a claim is filed, the damages and their causes are evaluated. The coverages included in the policy are also validated. Based on this, the insurance company will approve the repair of the damages or the financial compensation established in the policy's coverages.
- **Quality Assurance:** Insurance companies typically have a quality assurance service where clients can file complaints and claims. Discrepancies regarding the services provided or decisions made by the insurers are evaluated by an external entity seeking a certain degree of objectivity. The decisions previously made can be upheld or overturned.

The second phase focuses on the analysis from a technical and functional perspective of the operational architecture in a traditional system. The main processes are briefly detailed below:

- **Integrated Multi-Policy Management Process:** This constitutes the core of the company. It encompasses comprehensive management throughout the entire lifecycle of a policy. It includes the management of customer information, policy updates, and financial data such as scales, premiums, or insured capital.
- **Assessment Process:** Handles the entire lifecycle of claim assessment management. This process may be outsourced.
- **Quality Process:** Manages customer issues or complaints related to claims, policies, or assessment assignments.
- **Economic Management Process:** This involves the management of all financial aspects, such as payments, repair invoices, compensation, recovery management, and tax procedures. If there are outsourced processes, the billing for these services is also included.
- **Assistance Process:** Manages the entire lifecycle of a claim associated with a policy. This process may also be outsourced.
- **External Companies:** As mentioned earlier, certain processes may be outsourced. The company communicates with various entities using different methods (API, FTP, Batch) to exchange information, perform tasks, or carry out verifications. It should be noted that this can increase the existing relationships between entities, thereby potentially increasing system duplications. Additionally, it may lead to difficulties in terms of interoperability and system integration, data inconsistency, higher implementation and maintenance costs, etc.

Considering these processes, a typical traditional architecture is illustrated in Figure 2.

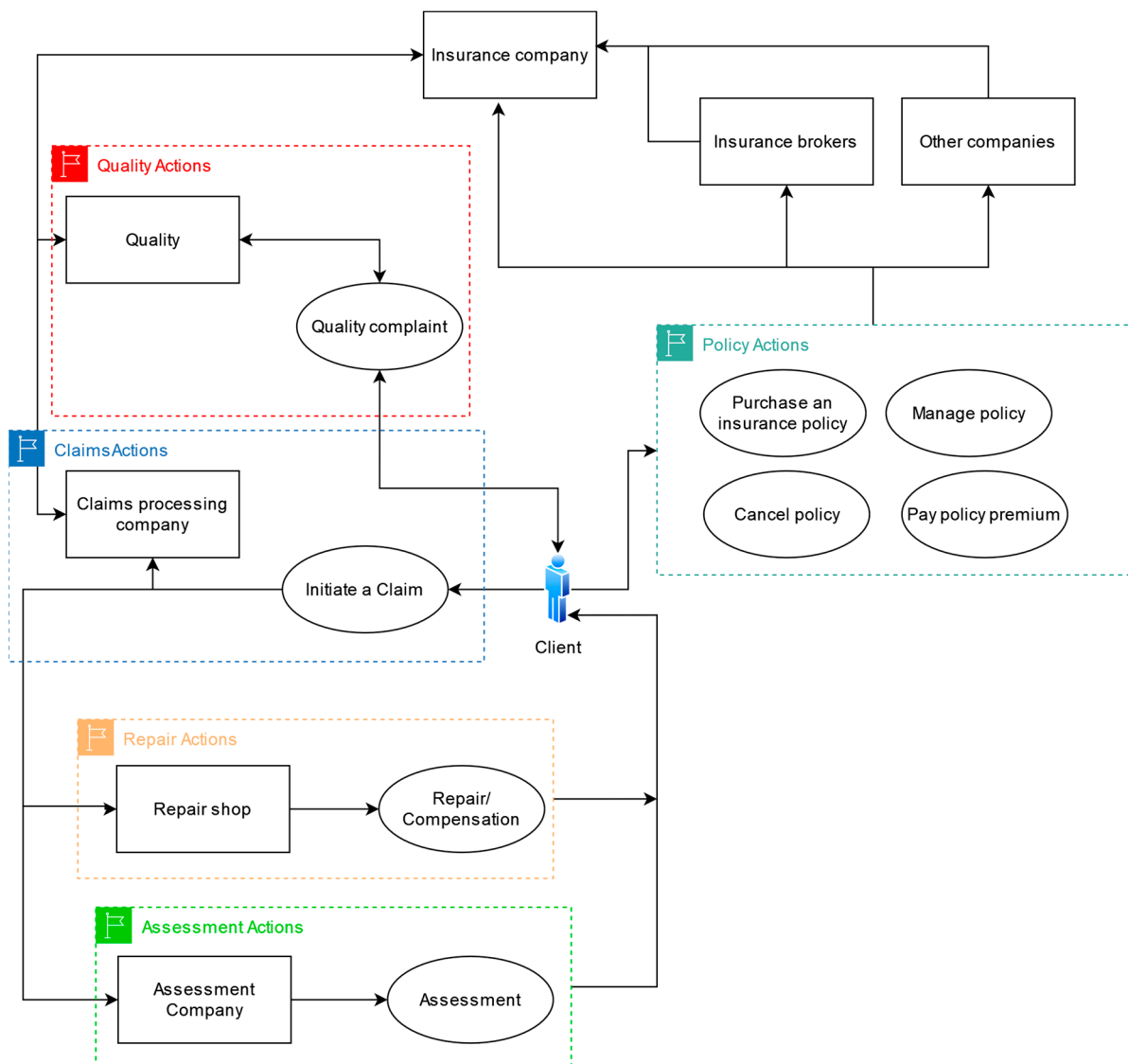


Figure 1. Client use cases.

Based on the traditional architecture (Figure 2), the following key issues can be identified:

- There is a significant amount of information that travels between different parts of the system, whether via APIs or files, which increases the risk to the integrity and consolidation of the data.
- There is a high level of operational redundancy as well as an excess of relationships between the different entities in the system. This leads to greater technical complexity, making maintenance and scalability more challenging. If a new participant needs to be added, all existing relationships would have to be replicated. For example, adding a document management entity would require replicating the entire operational flow of information exchange, processing systems, etc.
- Difficulty in maintaining effective fraud management: Due to the duplication of processes and continuous information exchange, it is necessary to accurately assess whether the cause of a claim is genuine, whether it aligns with what the customer has contracted, and to prevent possible undesirable alterations as previously mentioned.
- Difficulty in ensuring the integrity of the document management process: Ensuring the integrity of legal documents, such as contracts and invoices, is crucial because they must maintain their confidentiality and legal validity and guarantee the unique identification of the signer. These documents, which can be signed manually or

digitally, must be available to government, tax, or judicial authorities. The challenge lies in the multiplicity of actors and systems involved, complicating the consistency and coherence of the documents. Moreover, manual signatures are susceptible to errors and forgeries, while digital signatures require a secure and complex infrastructure. Guaranteeing the identity of the signer requires robust and costly methods. Finally, keeping these documents secure but accessible only to authorized personnel and ensuring compliance with legal and regulatory requirements adds an additional layer of complexity.

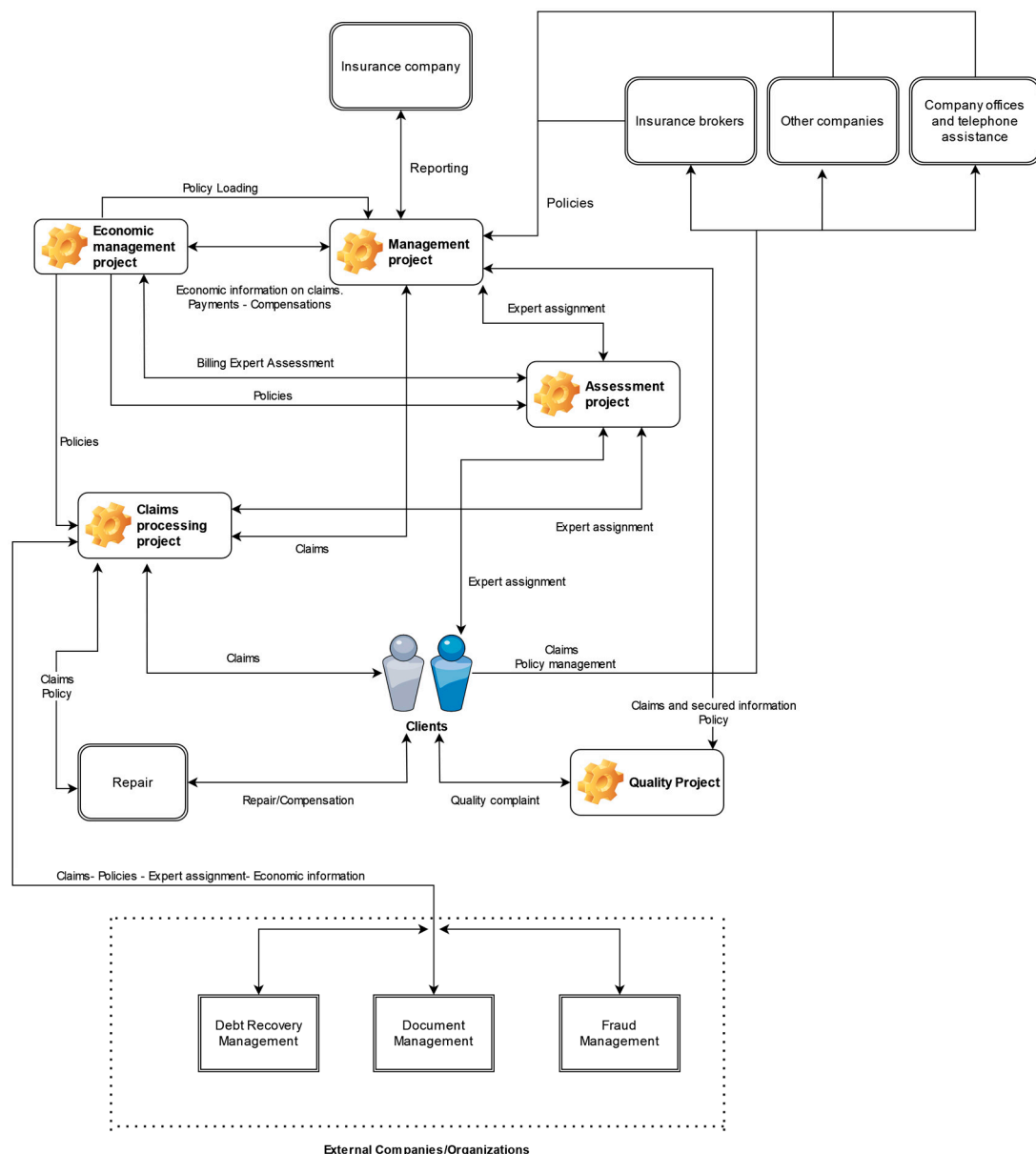


Figure 2. Architecture comprehensive management system current model.

We are dealing with an architecture that involves too many relationships between participants in order to integrate their different systems, leading to redundancies. To move towards a multiproduct policy model without adding further redundancies to the existing architecture, we propose evolving the solution towards a technologically different model. This new approach would allow us to adapt operational processes to the proposed standard policy model by utilizing a Blockchain network and smart contracts, where the key element is the policy itself.

3.3. Blockchain

We need an architectural model that allows us to eliminate the potential duplications and inconsistencies mentioned earlier. The approach we will take involves using Blockchain as the central core of the architecture. By leveraging this technology, we can integrate all processes and information management into a new decentralized architectural model, where all participants will have access to the same information.

A Blockchain solution applied to this business model can provide a swift response to various issues identified:

- **Duplication of processes:** By incorporating Blockchain as the central axis of the solution, all network participants can interact consistently using the same tools. This eliminates the necessity for proprietary infrastructure at each entity, resulting in cost savings and improved efficiency in administration, management, and processing.
- **Information Privacy:** By reducing the number of processes and necessary infrastructure, the likelihood of attacks on service availability or unauthorized access to data are also minimized. Furthermore, only authorized members of the network will have access to it, enabling them to perform transactions and interact with smart contracts.
- **Data Duplication:** Each participant in the network holds a copy of the distributed ledger where all information is recorded. This eliminates the necessity for components such as APIs or web services to manage this data. Entities will focus solely on managing and storing information on the blockchain. They will now access data remotely and securely, ensuring that information is not skewed or inconsistent across different systems, as they will all share the same copy of the data.
- **Validation Issues and Interactions:** By utilizing smart contracts, we can model the concept of insurance policies and the specific operations of the insurance business. This will enhance validation criteria, risk assessment, and report generation. Blockchain enforces consensus on these criteria, as all participants operate within the same network, ensuring uniformity. Smart contracts enable the definition of different states a policy and its associated claims go through, governed by a set of rules. These contracts can even be triggered automatically when certain events occur, simplifying the establishment of business-related rules.
- **Permissions and Certificates:** Given that the information model is distributed, each participant in the process can access different levels of information based on their access permissions. Digital certificates can be employed to ensure a high level of privacy, reduce errors in data handling, and improve fraud prevention efforts.

Additionally, Blockchain inherently possesses two crucial characteristics for the secure management of insurance policies.

- **Consensus:** This can be used to define the conditions under which a new client can purchase a policy, make changes to it, file a claim, or initiate a loss adjustment request. This is an automatic mechanism involving all system participants, significantly reducing the number of validations, processes, and communications required between stakeholders.
- **Immutability:** This is a standout feature of Blockchain. Once a transaction is added to the chain, it cannot be altered. Modifying an already recorded transaction requires a new transaction, with both being recorded. This ensures data integrity at all times, effectively eliminating information inconsistencies across different systems.

4. Solution Approach

This section outlines the proposed solution for managing policies in a multi-insurance company through the use of a private Blockchain network, leveraging the advantages of a Merkle tree-based implementation.

Once again, a use-case-based methodology [20] has been adopted to analyze the project requirements. Based on this analysis, the following functional and non-functional requirements have been defined.

This methodology has been essential in selecting the functional and non-functional requirements of the proposed solution. These use cases were identified through a detailed analysis of the various interactions between the insurance company and the other participants in the system: departments, clients, and third parties. From this analysis, the different necessities and specific challenges of these types of technical systems were extrapolated. This ensured that the requirements addressed real-world operational processes and the technical necessities to effectively integrate blockchain technology.

For example, the policy creation requirement arose from the necessity to standardize data across different insurance lines, as identified in the policy management use case. This led to the integration of Merkle trees and the use of smart contracts to generalize policies, ensuring consistency and reducing operational complexity across multiple products. Similarly, the requirement for blockchain network registration stems from the necessity for decentralized data management, ensuring that all relevant information about policies and claims is securely stored and accessible to authorized users on the blockchain, as highlighted in the claims submission and damage assessment use cases.

The management of roles and permissions was established based on the necessity to securely manage the various actors involved in the insurance process: insurance companies, policyholders, and external service providers. This requirement ensures that each actor has the appropriate level of access to interact with the system, perform specific tasks, and execute smart contracts based on their role within the infrastructure.

Additionally, non-functional requirements, such as system scalability and security, were driven by the necessity to handle larger loads and maintain data integrity as the system scales. These necessities were emphasized by the complexity of managing multiple interactions in various use cases, from policy issuance to claims settlement, where operational efficiency and security are paramount.

This approach ensured that the proposed solution was based on real-world practical necessities while also providing a clear roadmap for developing a scalable, flexible, secure, and efficient system.

The functional requirements are:

- **User Management:** The system must manage users involved, including the insurance company, policyholders, and third parties (e.g., processors, repairers). There should be a process for registering and authenticating these users within the system.
- The solution provides a decentralized system that allows user authentication through smart contracts, ensuring that only authorized users can access and operate within the blockchain network, guaranteeing data integrity and security.
- **Policy Creation:** A standardized data model is required, utilizing smart contracts to generalize policies related to different insurance lines.
- The solution provides standardization of policies through Merkle trees and smart contracts, which facilitates efficient management of multi-product policies and eliminates duplication of data and processes.
- **Blockchain Network Registration:** The information for each policy will be stored on the Blockchain network, providing a decentralized record of all information accessible to all system participants.
- The solution provides a decentralized ledger on the blockchain that ensures data consistency and eliminates duplication, ensuring that all participants access a single source of truth.
- **Blockchain Network Interaction:** Users must be able to execute smart contracts through a user application to perform operations such as purchasing policies, making payments, and filing claims.
- The solution provides automation of administrative tasks such as policy creation and claims management through smart contracts, reducing manual intervention, minimizing errors, and improving operational efficiency.
- **Role and Permission Management:** Users will interact with the system via smart contracts, performing transactions on the network. Permissions ensure that only

authorized users (according to their roles) can interact with smart contracts to perform specific actions, such as purchasing a policy, processing a claim, or making payments.

- The solution provides a decentralized permission management system, ensuring that only authorized users can interact with smart contracts and perform specific operations, such as creating policies or processing claims.

Non-functional requirements are:

- **Preserve System Security:** Ensure the integrity, authenticity, and availability of data within the system.
- The blockchain solution, together with Merkle trees, ensures the integrity and verifiability of data, reducing the risk of fraud through the immutable record of each transaction and the continuous verification of data authenticity.
- **Enhance Efficiency and Performance:** Improve the overall operational efficiency and performance of the system.
- The solution provides automation of key processes through smart contracts, significantly reducing processing times and improving operational efficiency.
- **Ensure System Scalability:** Make certain that the system can scale to handle increased loads and complexity as needed.
- The solution provides standardization of policies, and the use of Merkle trees allows the system to be scaled without increasing technical complexity, facilitating the integration of new insurance products without compromising performance.
- **Eliminate Duplications and Dependencies:** Remove redundant processes and reduce reliance on traditional systems.
- The solution eliminates redundancies by centralizing data on the blockchain, ensuring that all participants operate with the same information without duplication of processes.
- **Reduce Implementation and Maintenance Costs:** Lower the costs associated with deploying and maintaining this type of solution.
- The solution provides a reduction in technical complexity and automation of key processes that allow a significant reduction in operating and maintenance costs, especially in terms of scalability.
- **Improve User Experience:** Provide a smoother, more intuitive user experience to increase user satisfaction and engagement.
- The solution brings simplicity by allowing for more intuitive and efficient interaction through the user application.

Each of the functional and non-functional requirements is designed to address the specific challenges identified in the traditional insurance architecture discussed above.

4.1. Application of Blockchain in the Insurance Market: Selection of Blockchain Type and Technology

Once we understand the benefits of implementing Blockchain as a technological alternative in an integrated policy management system, we need to determine which type of Blockchain network is most suitable.

As is well known, Blockchain can be implemented as either public or private.

To make a decision on the type of network to use, two critical aspects have been considered: the efficiency and the privacy and access of the solution.

- **Efficiency:** This is a key aspect in business environments and becomes a critical factor during periods of high demand. Transactions on a public network tend to be slower due to the necessity to validate blocks across a larger number of globally dispersed nodes. This process not only consumes more time but also more computational resources. Additionally, the lack of control over participants can result in network congestion, especially during peak demand times, which can significantly slow down transaction processing. Private networks allow precise control over participants and minimize waiting times, ensuring smooth operations even under conditions of high demand. The ability to manage nodes and optimize resources ensures greater

efficiency, which is essential for keeping operational costs low and improving the speed and reliability of the service.

- **Privacy and Access:** In this business model, sensitive, confidential, and private data about clients (such as personal, health, and financial information) are handled. A public network is accessible to anyone, whereas a private network is hidden from non-participants. A private network allows the insurance company to reduce the risks of data exposure and ensure compliance with both internal and regulatory policies.

Considering the above factors, our proposed solution is based on a private Ethereum network. As is well known, Ethereum is a widely used network and highly trusted among the world's leading companies. It is open-source technology, highly accessible, and continuously evolving, with substantial support from a large community of developers. These characteristics make it suitable for our solution.

4.2. Standardization and Modeling of Insurance Policies

In order to standardize the multi-insurance policy, it is necessary to classify the different branches of insurance. This classification can vary depending on the operational modeling by companies. For this article, a decision was made to conduct a generic review as possible, leading to the following conclusion.

To generalize the policy, the following general classification is proposed:

- The classification will be based on two main categories of insurance: Life and Non-Life.
- Life insurance is understood to be those policies closely related to the survival or death of the insured. These can further be divided into two types:
 - Risk: Such as insurance for Death or Disability.
 - Savings: Pertaining to retirement plans.
- Non-Life insurance refers to all other types of insurance, also known as general or casualty insurance. This category encompasses a broader range:
 - Auto
 - Home
 - Liability
 - Health
 - Credit
 - Surety
 - Specific: Where various types of insurance can be grouped, such as travel, breakage, or theft of devices, etc.

Building on this initial classification, the next step is to analyze each type of insurance individually, identifying the commonalities and isolating the specific characteristics of the elements that form part of each corresponding policy.

The first elements to identify are the participants; in this case, the modeling has been carried out considering the following roles: Policyholder, Insurer, Insured, and Beneficiary.

Next are the policy coverages, which specify the events or potential risks that will be covered by the insurance. For example, in the case of auto insurance, the coverages might include glass breakage, fire, or theft. These coverages can be grouped into products. Continuing with the previous example, this could refer to comprehensive insurance. A single policy may contain one or multiple products.

Next, the policy conditions begin to be defined, which can be divided into the following categories:

- **Particular Conditions:** These are unique to each policy, such as the effective date, its validity, or the resulting premium.
- **General Conditions:** These are common to all policies and typically include information such as exclusions, methods of payment for claims, etc.
- **Specific Conditions:** These apply to each particular type of insurance.

With all this information, a model for a generic policy is proposed, applicable to various types of insurance in a standardized and general manner. This model ensures scalability and maintenance by being considered as a set of objects in itself rather than a single template as traditionally used by different companies.

In Figure 3, which represents the modeling of an insurance policy, the different data groups that make up the standardized model we have developed can be observed. The final model comprises information specific to the insurance company, the policyholder's data, the contracted products, and the elements being insured. It also includes particular, general, and specific conditions that help define the nature of the policy. This generalized model is sufficiently flexible and comprehensive to cover the various types of existing insurance, as it allows for the easy addition of new types. Each structural element is designed to capture all the variations and requirements of any policy.

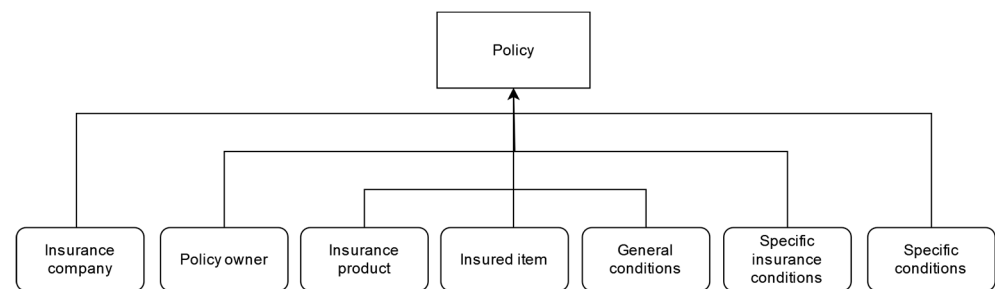


Figure 3. Policy modeling.

For example, to add a new insurance policy for electric scooters, a new type of insurance can be defined, with the scooters described as an Insured Asset, including their specific characteristics such as brand, model, value, and age. The particular and general conditions can be adapted to include specific coverages for accidents and theft, adjusting the relevant exclusions, definitions, and procedures. This model also allows for the inclusion of particular conditions, such as the premium and payment method, ensuring that all aspects of the policy are recorded and managed in a uniform and standardized manner.

All of this is achieved without altering the already established policies. This model provides a flexible and adaptable structure that allows new types of insurance and their specific characteristics to be added without modifying the structure of existing policies. As a result, we can ensure smooth integration without causing disruptions to the system while also facilitating the underwriting and management of any type of insurance policy.

4.3. Anchoring Protocol: Merkle Trees

The most immediate anchoring mechanism to the network would be through a smart contract. In it, the complete policy model would be represented directly. The main problems we might encounter with this initial approach are:

- **Size and Complexity:** Inserting large volumes of data directly can lead to performance and efficiency issues, affecting execution speed.
- **Model Changes:** Updating or correcting data in a part of the contract can be complicated and costly. This is because each modification requires a new transaction, which, with this model, involves the entire dataset of the contract or policy.
- **Code Security:** The larger the contract, the more attention must be paid to code security. Including errors or vulnerabilities in the contract's logic can create serious issues in policy management.
- **Data Queries:** Performing complex queries and analyses on large volumes of data stored directly in the contract requires processing all of it. It would be more efficient to only act on the necessary data.

To avoid this situation, we propose using Merkle trees as the insertion mechanism into the Blockchain network.

Practical approaches have been made where Merkle trees are used as a data protection mechanism for medical records instead of a full-scale Blockchain approach [21], as well as implementations where they serve as the insertion mechanism for training models for fault detection in IoT [22].

Merkle trees are an efficient data structure that provides a secure mechanism for verifying the integrity of large datasets used in cryptography and distributed systems. They were designed by Ralph Merkle in 1979.

Merkle trees are binary hash trees where each leaf node represents a piece of data, and each internal node represents the concatenation of the hash values of its child nodes. The root of the tree, known as the Merkle root, represents the unique cryptographic summary of all the data stored in the tree.

The verification or data search process when using Merkle trees is conducted as follows:

1. Obtain the Root Hash: This represents a summary of all the data.
2. Calculate Intermediate Hashes: To verify a specific piece of data, its hash is calculated and compared with the hash stored in the tree's leaf node. It is not necessary to process all the data.
3. Trace the Verification Path: Starting from the data's hash, trace the path from the leaf node to the root hash, calculating the intermediate hashes at each step.
4. Compare with the Root Hash: Finally, the root hash calculated from the verification path is compared with the root hash stored on the Blockchain. If they match, the data are valid and has not been tampered with.

Each time a transaction affecting a policy is made, the corresponding Merkle tree is updated to reflect the changes. Authorized entities, such as insurance companies, clients, and processors, can verify the complete integrity of the data by checking the Merkle root stored on the blockchain.

With this in mind, an insertion mechanism into the Blockchain network has been implemented based on Merkle trees instead of directly embedding the model of each policy within the smart contract. The advantages of this approach are as follows:

- Flexibility and Scalability: By representing each specific piece of policy data—such as the type of insurance, policy status, policyholder, etc.—through a Merkle tree, the system can easily adapt to any type of policy without limitations. Only the necessary data to represent that policy are stored in the tree. In contrast, with a standard smart contract, all types of data must be integrated from the outset, regardless of whether they are used.
- Granularity: More precise and detailed verifications can be performed on the data of a specific policy without the necessity to consider all of the client's policy data, thereby improving the system's transparency and reliability.
- Integrity: With smart contracts, detecting a hash discrepancy indicates that something has changed, but it does not reveal what has changed. Merkle trees, however, allow you to detect exactly which data point has been altered.
- Computational Cost: While data insertion with Merkle trees is somewhat more complex due to the necessity to calculate multiple hashes, the search operation, which is more frequent, has a significantly lower computational cost compared to smart contracts.
- Ease of Maintenance: This structure simplifies the maintenance and management of data, as changes are directly reflected through the verification of tree nodes. This facilitates the tracking and management of policy changes over time.

It is important to emphasize the ability to detect exactly where a change has occurred in a policy in an absolute and unambiguous manner. This allows us to identify the specific data point that has changed. When a piece of data in the policy is modified, its hash changes, which then propagates upwards, altering the root hash as well. To determine that the policy has changed, it is sufficient to compare the hashes; additionally, we can identify where the change occurred without needing to review all the data. In contrast,

when storing all data directly in the smart contract, change verification becomes more complex and less efficient, as it requires comparing all stored data.

Furthermore, using a smart contract for data modeling always necessitates a static and comprehensive model capable of representing any type of policy. With Merkle trees, we obtain a dynamic model that represents only the necessary data for each created policy, making it more suitable for the proposed standard policy solution. This approach provides a flexible and efficient representation, avoiding system overload with unnecessary data.

This proposal offers an effective solution in the scenario of multiproduct policies that we have presented, as it helps to abstract the solution by enabling an efficient and scalable data structure. Each policy is represented by a unique Merkle tree before being inserted into the network. This allows network participants to quickly verify whether the stored data are authentic and has not been altered. It also ensures greater confidence in the accuracy of the information associated with each insurance policy, thereby reducing the risk of disputes and conflicts.

Additionally, the use of Merkle trees as an anchoring mechanism to the network allows us to improve the scalability of the proposed architecture compared to traditional models. Let's consider an example where our insurance company manages two policies: one for life insurance and another for automobile insurance. In a traditional system, each of these policies is managed independently, using, for instance, traditional databases, independent data models, and different processing systems. This means that when adding a new type of policy, such as health insurance, significant changes must be made across every system involved. These changes would impact everything from the service core to the front-end applications and third-party systems. It would also be necessary to incorporate new business rules to handle different coverages, as well as to add new data import flows and verification processes. This approach results in data redundancy and high operational costs due to the technical complexity of updating multiple systems without compromising data integrity.

With our blockchain-based model, utilizing Merkle trees as the anchoring mechanism based on the standardized policy model, the addition of a new policy, such as health insurance, does not require reconfiguring the existing infrastructure. Instead, each multiproduct policy is represented by a Merkle tree, allowing new policies to be managed with the same data structure. The remaining trees, and consequently the stored information, remain intact. The information for each policy is stored using a Merkle tree, which allows the verification of that information without needing access to the full data content while also identifying the exact change. This ensures that even when new policy types are added, the system maintains its efficiency and scalability without affecting the existing policies.

In this way, the migration of data from the traditional system to the new blockchain-based model can be carried out efficiently through automated scripts. These scripts transform the data of existing policies into the new standardized model, minimizing errors and accelerating the migration process.

Finally, the advantages of this approach include reduced maintenance costs without interrupting production operations or compromising data integrity.

Two possible strategies can be considered for incorporating this data structure into the proposed architecture:

1. Each leaf node represents a complete policy: In this scenario, the root of the tree within the Blockchain network would represent a client. As depicted in Figure 4, each leaf of the tree represents a complete policy held by that client, who may have one or several policies at the same time. In this case, conflict resolution and verification are swift—by checking the hash of the tree root, we can detect changes in the client's entire set of policies, or by checking the hash of each leaf node, we can detect changes in a specific policy. In terms of storage efficiency, this approach is advantageous when the policies are small and each client holds multiple policies simultaneously.

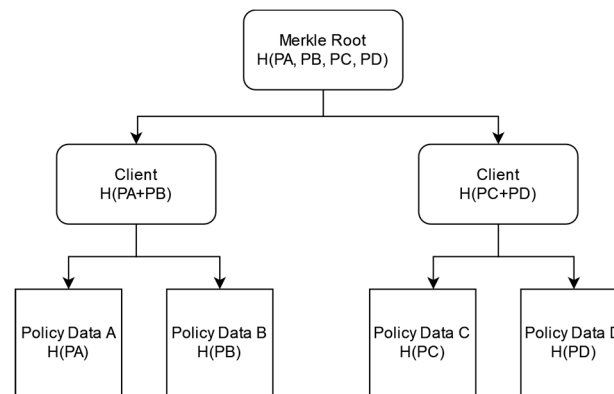


Figure 4. Leaf node as policy.

2. Each leaf node represents a policy data point, and each root represents a complete policy: In this second option, complete policies are recorded on the Blockchain rather than clients. Each leaf node of the tree represents a specific data point of the policy, providing greater granularity to the model. In other words, each client will have as many trees associated with them as the number of policies they have purchased. This model is depicted in Figure 5. In this case, verification can be more thorough, allowing the precise location of any changes to be identified. This alternative is more scalable when the number of policies per client varies significantly, as it only requires adding or removing nodes for each policy depending on the data needed.

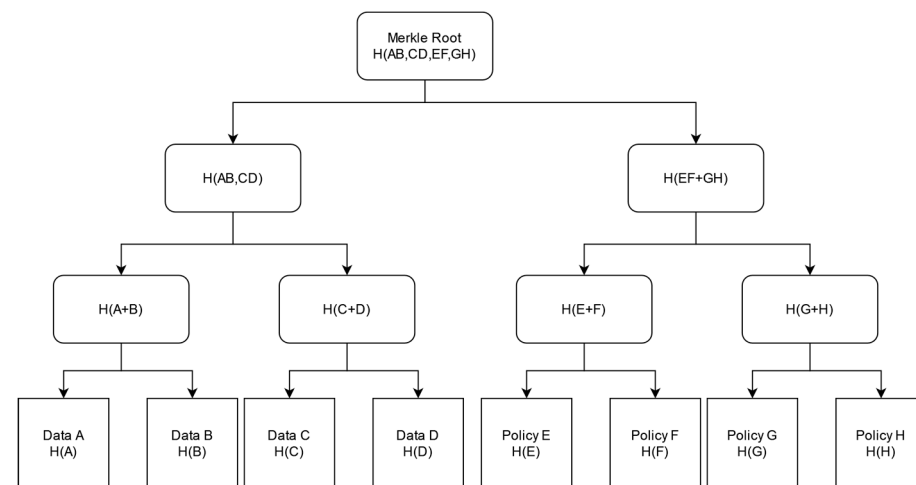


Figure 5. Tree root as a policy.

The strategy ultimately chosen is the second one, where each leaf node represents a specific data point of the policy, and the tree root represents a complete policy. This decision is based on the greater granularity provided by this option. It offers increased flexibility in managing and verifying individual policy data, allowing for the detection of changes at the exact point where they occur. The first strategy allows for the comparison of only the complete hash of each policy, detecting changes but not identifying the specific data point where the change occurs.

In client management, having each complete policy as leaf nodes would require a static data model to account for all possible options based on the type of insurance being handled. This approach would reduce adaptability and efficiency when adding new policies, as it would necessitate changes to these predefined models. In contrast, the selected strategy allows for the addition of new policies without altering the rest, as it is a fully dynamic solution. By leveraging the standardized policy model, we will add only the data that is truly necessary to the tree.

Additionally, considerations such as the size of the policies and the frequency of data updates have been taken into account. These factors directly impact the scalability of the solution. With the chosen option, each tree represents a single policy, which facilitates the organized management of large volumes of data.

In both solutions, the primary element is always the policy. The inclusion of the client entity in the first option simply reflects a different way of organizing the information. This approach is more similar to the initial method of using only smart contracts, where a predefined data model is required.

Continuing with the previous example of adding insurance policies for electric scooters, let's suppose a client already has two existing policies, one for home and another for health. Each of these policies has its own Merkle tree already registered on the Blockchain. To insure a scooter, a new Merkle tree would simply be created for this policy without the necessity to modify the existing trees. This is possible because each Merkle tree represents an independent policy. This approach contrasts with directly using smart contracts, which would require modifications or updates at the contract level to manage these changes. It also differs from the first proposed Merkle strategy, where modifying or adding a policy would require recalculating and updating a larger tree that groups all of the client's policies.

4.4. Blockchain Network Architecture

Based on the analysis conducted, the proposed architecture is illustrated in Figure 6.

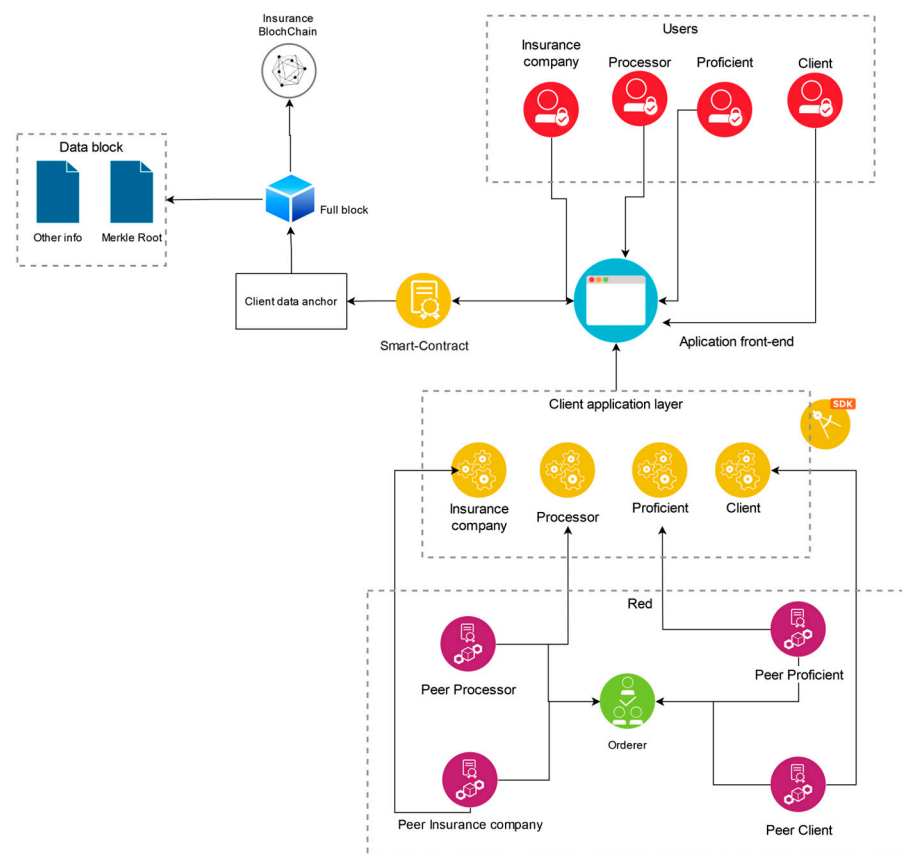


Figure 6. Architecture design Blockchain.

The architecture consists of several distinct components. On one side, there is the Blockchain network, which includes a defined set of participants: the insurance company, processors, appraisers, and clients. These network participants can interact visually and graphically through a set of web applications, which are, in turn, connected to a client application layer that encapsulates the available operations for interacting with the Blockchain network.

On the other side, we have the smart contract, which facilitates the insertion of the Merkle tree root containing each policy's content into the Blockchain network. Additionally, it provides the necessary verification mechanisms through hash calculation and comparison. The smart contract is responsible for transaction control within the network and contains the business logic related to policies, such as data validation, policy conditions, and the execution of contractual clauses. This ensures that insurance rules are applied automatically and consistently.

Both clients and insurance companies can create insurance policies through the provided APIs, which will interact appropriately with the Blockchain network via transactions and the smart contract.

When a policy is created, it follows the previously standardized data model, from which a Merkle tree is generated for each policy. This tree will be used through an insertion protocol into the Blockchain network as a mechanism for verification and dispute resolution on the blockchain.

The process is as follows: Each individual piece of policy data are transformed by obtaining its hash value. Each pair of leaf nodes is then combined, and the hash value of their combination is computed using the same hash function. This process is repeated until the root of the tree is reached, resulting in a final hash.

Let's consider a very simple example to illustrate how the root of the tree is composed for a policy with four data points. As illustrated in Figure 7, the hashes are first computed for the leaf nodes, and then each pair of nodes is combined to generate their combined hash. This process is repeated until the root node is reached, where we have the final hash: 84508d01e777c847d626d979309a0bd9 for our example.

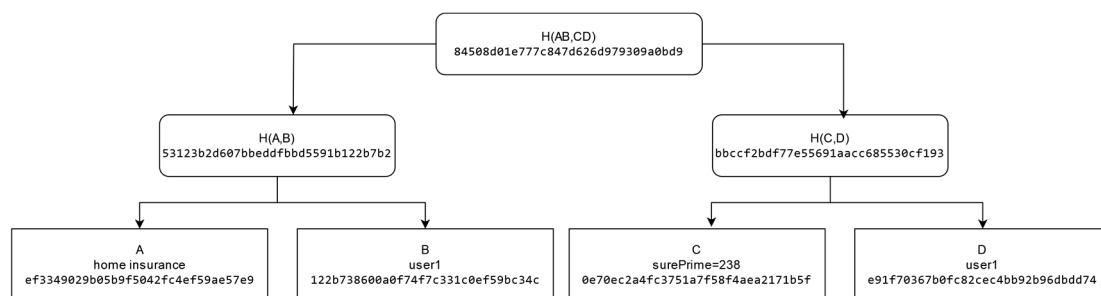


Figure 7. Merkle example.

Once the tree is completed, a block will be created containing the unique identifier of the policy, the entire tree, and a digital signature from the entity. This block is then added to the Blockchain network.

Referring to Figure 6, an example of an operation could be: a system user, such as a client, accesses the available processes to perform operations. They could create new policies, update existing ones, file a claim, etc. This web interface connects to the Blockchain network through the application layer built using the necessary frameworks and SDKs. This setup allows the user to operate via the smart contract based on Merkle trees. Suppose the user is creating a new policy. Through the web interface, the user fills in all the required information fields, which are then converted into a Merkle tree as described earlier and subsequently inserted into the Blockchain network after invoking the smart contract. Each complete block is inserted as represented in Figure 8:

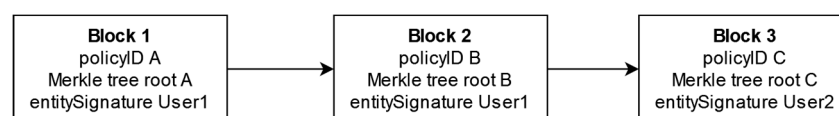


Figure 8. Blockchain example.

5. Solution Implementation

This section details the implementation of the architecture we propose as a solution to the problem at hand.

5.1. Technology Stack

As previously mentioned, the selected Blockchain network is private and based on Ethereum. Therefore, we have chosen to use Hyperledger Besu for the final implementation. The decision to use this technology stack is based on the capabilities offered by each of the tools and software components. The goal is to build an execution environment that is efficient while also ensuring robust and secure transaction management. Additionally, we need a smooth and user-friendly interaction between users and the Blockchain network.

As illustrated in Figure 9, the following section shows the set of technologies used and how they integrate to form the complete solution.

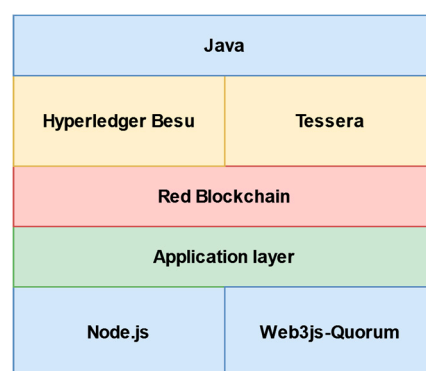


Figure 9. Technology stack.

- **Node.js:** This serves as the primary runtime environment for our web services, APIs, and applications. The choice of Node.js is based on its ability to execute JavaScript code without the necessity for a browser, which is crucial for developing server-side applications.
- **Web3js-Quorum:** This library integrates with Node.js and allows interaction with the blockchain. Web3js-Quorum extends the functionality of the standard web3.js library, providing compatibility with the JSON-RPC specifications of GoQuorum and Hyperledger Besu. It also handles privacy groups and facilitates the creation and submission of transactions to the blockchain.
- **Hyperledger Besu:** This acts as the Ethereum client within our blockchain network. Hyperledger Besu is responsible for managing transactions, validating blocks, and maintaining the blockchain.
- **Tessera:** Working alongside Hyperledger Besu, Tessera manages private transactions within the network. Tessera ensures that confidential transactions are handled securely and efficiently, providing privacy on the blockchain network.
- **Java:** Java is included as an essential prerequisite for running Hyperledger Besu. This runtime environment is necessary to ensure the operability of the Ethereum client on the network.

The diagram illustrates how each component of the technology stack interrelates and contributes to the overall functioning of the blockchain network. Node.js provides the runtime environment, on which Web3js-Quorum facilitates interaction with the blockchain. Together, these form what we refer to as the application layer.

Hyperledger Besu and Tessera are responsible for managing the blockchain and private transactions, respectively. Collectively, these technologies constitute the Blockchain network layer.

Finally, Java is an essential requirement for the operation of Hyperledger Besu.

This modular and well-defined architecture ensures an efficient and secure implementation of the Blockchain network.

5.2. Blockchain Network Implementation

5.2.1. Creating Network Nodes

We have developed an initial Blockchain network that includes four different entities. Each of these entities corresponds to a node within the network:

- Insurer: Represents the primary node of the network and is considered the 'owner' of the deployed system. It also acts as an additional participant in the network.
- Processor: This node represents the network entity responsible for receiving and managing claims created by the client or policyholder within the insurance company.
- Appraisal: This node represents the appraisal phase that may occur during a claim. It functions as another company within the ecosystem of an insurance company.
- Client: Represents the node that acts as the policyholder or customer of the company. This node is responsible for purchasing the policy and generating claims.

5.2.2. Network Configuration

The initial network configuration is performed through a single JSON file called *ibftConfigFile.json*. Within this file, two key sections are configured. First, the genesis configuration of the network, which is used to generate the first block of our Blockchain. Second, a Blockchain configuration section, where the number of nodes in the network is specified, along with the necessary key pairs.

As shown in Figure 10, the most notable elements of this configuration file are:

- chainId: This is the identifier assigned to the network.
- IBFT 2.0 defines the properties of the selected consensus algorithm, which in this case includes:
 - blockperiodseconds: The minimum block time, in seconds.
 - epochlength: The number of blocks after which all votes are reset.
 - requesttimeoutseconds: The timeout period, in seconds, for each consensus round before a round change occurs.
- gasLimit: Represents the maximum gas limit for each block in the Hyperledger Besu network. In our architecture, we operate with a free-gas network, so we set this parameter to a very high value to effectively ignore it.
- Difficulty: This value is used to control the rate of new block creation, thereby maintaining the network's security. In our network, it is set to the hexadecimal value 0x1, which equals 1 in decimal. This means that the mining difficulty within the network is relatively low, making it easier to mine new blocks.
- Alloc: This is used to allocate funds to specific accounts when initializing the Hyperledger Besu network. It allows balances to be set for accounts before any transactions take place on the network, enabling test operations.

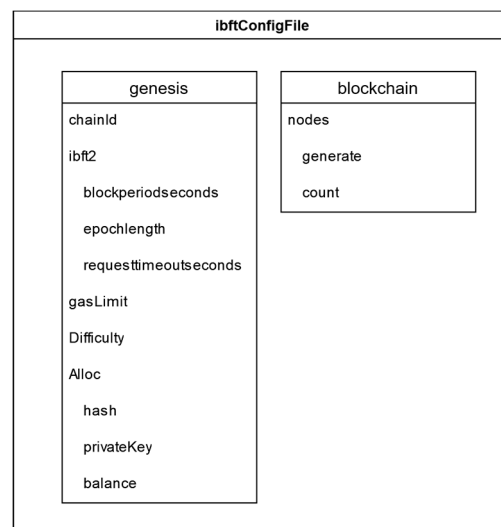


Figure 10. ibftConfigFile.

5.2.3. Node Keys and Genesis File

Using the previous configuration file, you can generate the set of public/private keys for each node in the network, as well as the genesis file.

To execute this process, simply run the following command:

```
besu operator generate-blockchain-config--config-file = ibftConfigFile.json--to = networkFiles--private-key-file-name = key
```

With this Besu command, two actions are performed:

- **Genesis File Creation:** It generates the *genesis.json* file based on the properties defined in the previous configuration file. Additionally, the *extraData* property is added, indicating that the four nodes in the network will serve as validator nodes.
- **Node Directory Creation:** A directory is created for each node, named using the node's address, and it contains the public and private keys for each node.

The final result is a structure similar to the one shown in Figure 11:

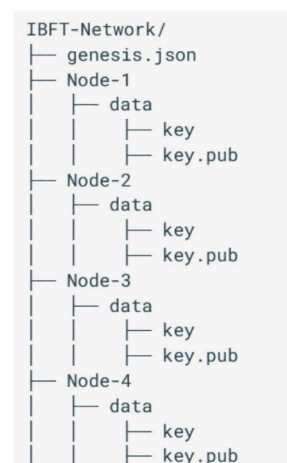


Figure 11. Besu project structure definition.

5.2.4. Privacy Groups

The next step is to add the necessary configuration to create privacy groups. This ensures that private transactions are allowed only between the members of that group. This is achieved through the Tessera component.

Tessera is a private transaction manager specifically developed for enterprise Blockchain environments where a high degree of privacy and confidentiality is required. It is designed to work with Hyperledger Besu, providing a robust system for handling private transactions on either a public or private Blockchain network.

In this context, Tessera allows us to create and manage the privacy group within our architecture. We have defined a set of nodes authorized to view and participate in the network through private transactions.

The operation of Tessera can be observed in Figure 12.

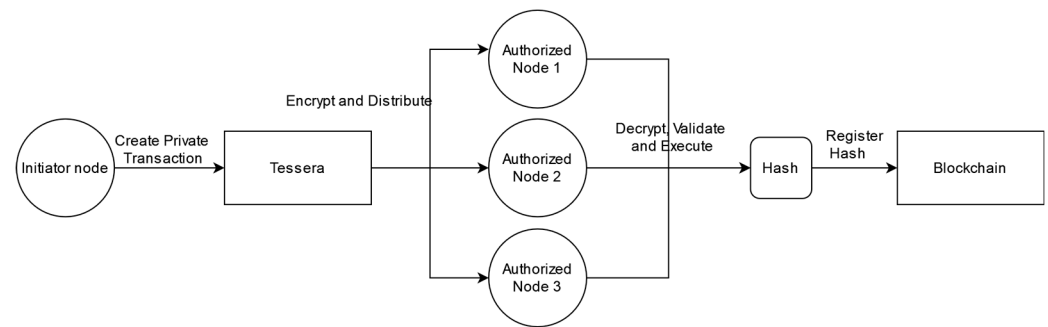


Figure 12. Tessera operation.

To configure Tessera, we essentially need to execute the following command within each of the nodes:

```
tessera-keygen-filename nodeKey
```

The result of this process is the generation of a public/private key pair for each node, which will be used during the creation of private transactions within the network.

5.2.5. Tessera Configuration

To include Tessera within the Blockchain network, we simply need to add one more configuration file called *tessera.conf*. You create one file for each node in the network. This JSON-format file allows you to configure the following aspects of the network:

- **Mode:** Specifies the operating mode. In this case, Orion is used to ensure compatibility with previous versions of the network.
- **useWhiteList:** Allows the specification of a whitelist for transaction control. With the value set to false, all incoming transactions are processed.
- **jdbc:** Contains the database configuration for storing information.
- **serverConfigs:** Allows the configuration of the path and port associated with each node in the network to be invoked.
- **peer:** Contains information about other nodes in the network with which Tessera can communicate.
- **keys:** Defines the private and public keys used for encrypting and signing transactions.

The representation of a Tessera node with respect to its main components can be seen in Figure 13:

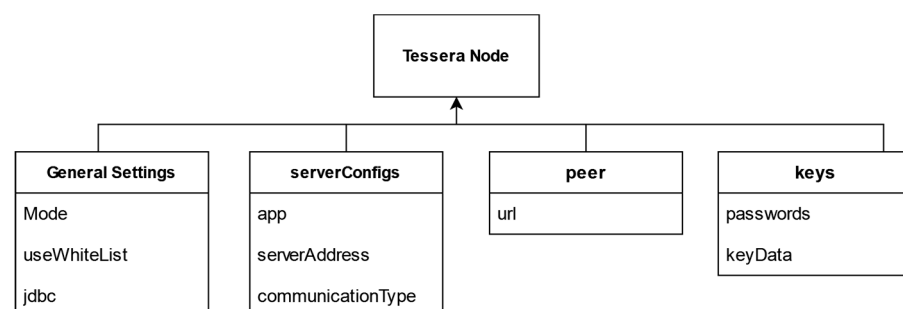


Figure 13. Tessera node configuration.

5.3. Anchoring Protocol Implementation

The implementation of an insertion protocol into the Blockchain network is essential, given the decision to use Merkle trees for representing each policy. As previously defined, the model used involves each Merkle tree representing a complete policy, with each leaf node representing a specific piece of policy data.

We have defined an insertion protocol whose primary objective is the construction of a Merkle tree based on the data of each policy. The root of each tree will be anchored to the Blockchain network. This ensures the integrity and auditability of each client's data. Moreover, it is crucial for verification purposes, particularly in resolving disputes between the insurance company and other network participants, especially the clients.

The insertion protocol has been defined as follows:

- **Data Decomposition:** The data that make up the policy are broken down into individual elements. Each element is treated as a leaf node of the tree.
- **Tree Generation:**
 - Each leaf node undergoes a hash function using the SHA-256 algorithm.
 - The resulting hashes are grouped in pairs and are again subjected to the same hash function. The results form the nodes of the next higher level.
 - This process is repeated iteratively until the root node is reached.
- **Insertion into the Blockchain Network:**
 - The root of the tree, which contains a hash summary of the entire policy content, is stored on the Blockchain through a transaction.
 - The root of each tree serves as the insertion point or reference for the data of each stored policy.
- **Verification:**
 - Each specific data point in a policy can be verified by tracing the complete path from the corresponding leaf node to the root.
 - It is also possible to verify the overall data integrity by comparing the computed root with the stored root.

Algorithm 1 outlines the operation of the insertion protocol into the Blockchain network:

Algorithm 1. Merkle Tree Generation and Anchoring

```

1: Policy ← insurance policy data
2: /*Client executes*/
3: struct {policyData, merkleRoot} policyHash
4: for each data ∈ Policy do
5:     leafNodeHash ← hash(data)
6:     leafNodesList.append(leafNodeHash)
7: end for
8: merkleRoot ← buildMerkleTree(leafNodesList)
9: anchorToBlockchain(merkleRoot)

10: /*Merkle Tree Construction */
11: buildMerkleTree(leafNodesList){
12:     while leafNodesList.length > 1 do
13:         newLeafNodesList ← []
14:         for i from 0 to leafNodesList.length1 step 2 do
15:             if i + 1 < leafNodesList.length then
16:                 node ← hash(leafNodesList[i] + leafNodesList[i+1])
17:             else
18:                 node ← leafNodesList[i]
19:             end if
20:             newLeafNodesList.append(node)
21:         end for

```

Algorithm 1. *Cont.*

```

22:     leafNodesList ← newLeafNodesList
23:     end while
24:     return leafNodesList[0]
25: }
26: /*Blockchain Anchoring */
27: anchorToBlockchain(merkleRoot){
28: transaction ← createTransaction(merkleRoot)
29: sendTransactionToBlockchain(transaction)
30: }

31: /*Policy Data Verification */
32: verifyPolicyData(data, merkleProof, merkleRoot){
33:     leafNodeHash ← hash(data)
34:     calculatedHash ← leafNodeHash
35:     for each partialHash ∈ merkleProof do
36:         if calculatedHash == partialHash.left then
37:             calculatedHash ← hash(calculatedHash + partialHash.right)
38:         else
39:             calculatedHash ← hash(partialHash.left + calculatedHash)
40:         end if
41:     end for
42:     return calculatedHash == merkleRoot
43: }

```

To construct the Merkle tree, the first step is to go through all the elements of the policy and calculate the corresponding hash for each. These calculated hashes are then inserted into the *leafNodeHash* list.

Using the *buildMerkleTree* function, the tree is constructed. In this process, each pair of hashes from the previous list is combined, and their combined hash is calculated. This process is repeated until only a single hash remains, which will be the root node of the Merkle tree.

The next function, *anchorToBlockchain*, involves creating a transaction with the tree root. This transaction is then sent to the Blockchain network to be stored.

Finally, the *verifyPolicyData* function allows for the verification of a policy's data. First, the hash of the data to be verified is calculated. Then, using a list of partial hashes, the tree root is calculated. If the calculated root matches the root stored on the Blockchain network, we can determine that the data are valid and has not been altered.

The entry of policy data, hash generation, and Merkle tree creation are all performed at the level of the user DApp. Once all operations have been completed in the DApp, the smart contract is invoked to record the tree root on the Blockchain network. Additionally, each participant in the network can verify the policy data directly from the DApp as a preliminary step before performing any operations. Additionally, each participant in the network can verify the policy data directly from the DApp as a preliminary step before performing any operations (Appendix A).

This entire process is illustrated in Figure 14:

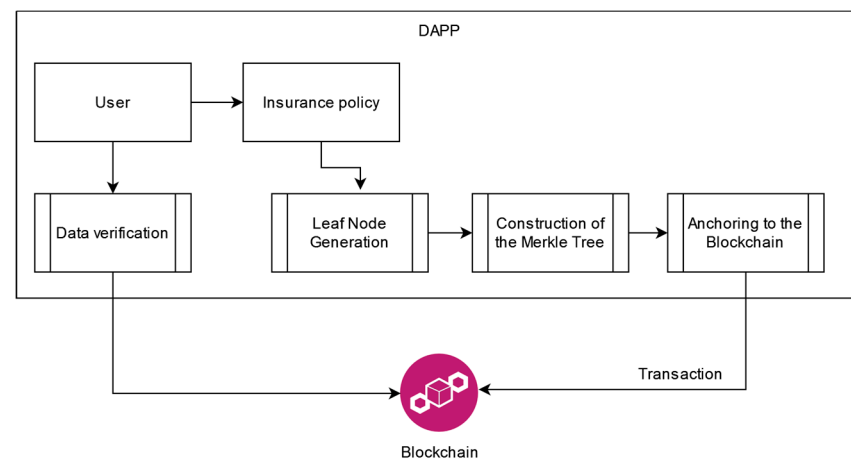


Figure 14. Anchoring protocol.

5.4. Smart Contract Implementation

The chaincode or smart contract is the component of our Blockchain network that allows us to interact with the network. By generating new transactions, we can add new information to the blockchain.

The programming language used to create the smart contract in Ethereum is Solidity. Solidity is a high-level, object-oriented programming language. One of Solidity's strengths is its straightforward syntax and its ability to compile into bytecode. This generated code will be executed on the Ethereum network.

The first step we took was to model the new standardized policy model in programming code. To accomplish this, we created the UML diagram represented in Figure 15:

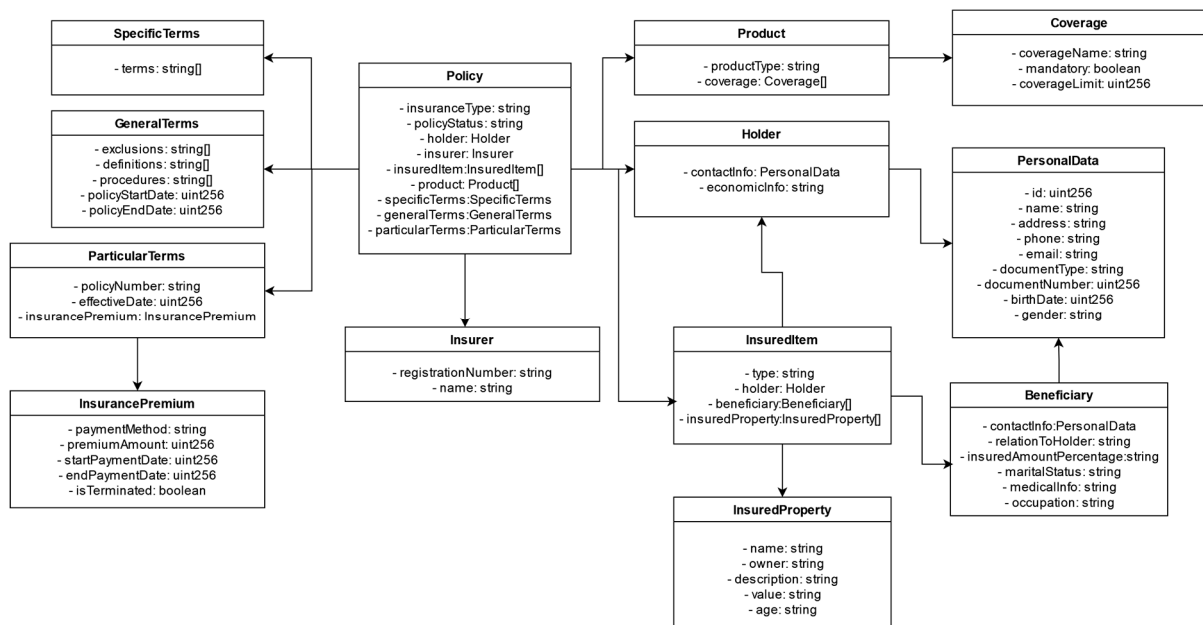


Figure 15. UML class diagram modeling multi-product insurance policy.

With the new policy model established, it is time to construct the smart contract. As previously mentioned, the policy is integrated into a Merkle tree, and the root of that tree will be the element recorded in each block of the Blockchain. The UML model of the smart contract can be seen in Figure 16:

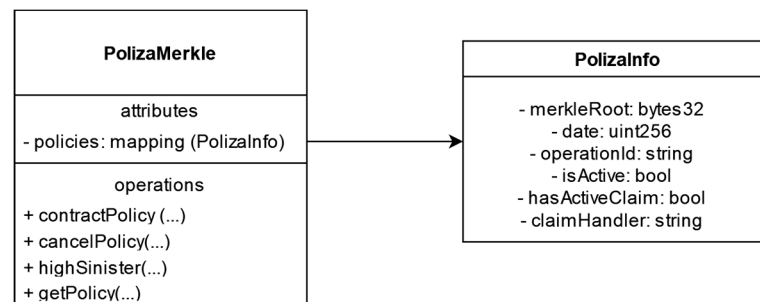


Figure 16. UML smart-contract representation.

As can be seen, the smart contract contains a set of attributes and a set of operations:

- **Attributes:** These represent each policy stored on the Blockchain. They include data such as the operation date and operation identifier, and these data will be used for controlling transactions on the network. Additionally, there are indicators for an active policy and an active claim. The most important attribute is the Merkle root, which contains all the data of the policy.
- **Operations:** These represent the available operations within the smart contract. In this particular case, the most common operations for an insurance company have been created.

Following this, Figure 17 presents the sequence diagram, which illustrates all possible interactions with the smart contract, including policy creation, cancellation, claim filing, and retrieving policy details:

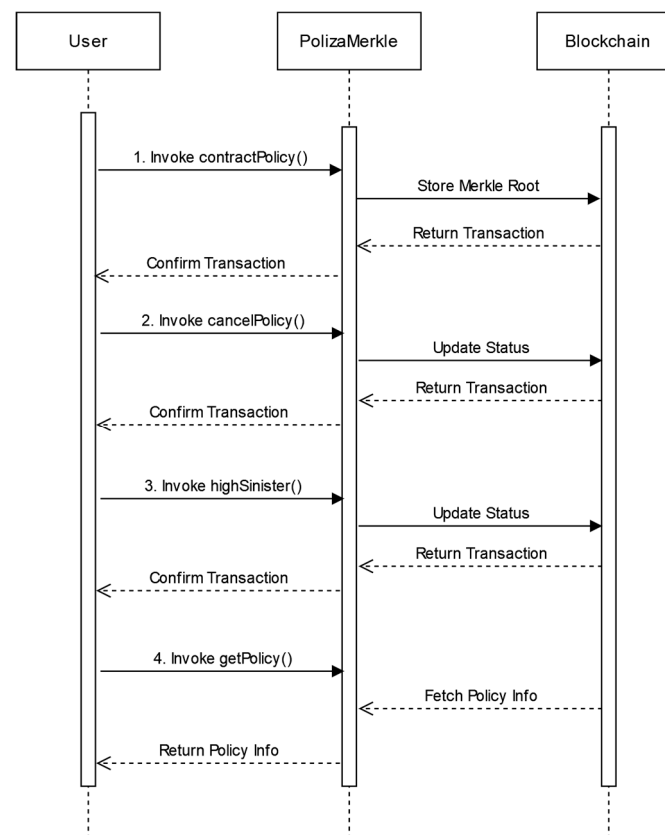


Figure 17. Smart-contract operations sequence diagram.

The final step in implementing the smart contract is compiling and generating the bytecode and ABI (Application Binary Interface) for deployment on the Blockchain network.

The compilation phase is performed using a tool like solc, which is a Solidity Compiler. The result of this operation will be the bytecode and the ABI.

- **Bytecode:** This is the compiled binary code that can be executed by the Ethereum Virtual Machine (EVM) and is what is actually deployed on the network.
- **ABI:** This is a specification that describes how to interact with the smart contract, detailing the available functions and data. It acts as an interface between the contract and any application that wants to interact with it, such as our DApp.

For deploying the contract, we used web3.js, creating a custom script for interacting with the Blockchain network. This script sends a transaction, and the result is the address of the deployed contract, which will be used for all subsequent interactions from the DApp.

5.5. Application Implementation

To enable users and clients to interact with the Blockchain network, we will implement a DApp (Decentralized Application). These are applications that run on peer-to-peer networks. The key advantage of such applications is their ability to offer greater transparency and security. This makes them particularly suitable for use in Blockchain systems.

We have implemented the application layer using Node.js. The choice of this technology is justified by the ease of using JavaScript as a programming language. This allows us to utilize the web3.js library to interact with the previously created smart contract and conduct transactions within the Blockchain network.

Web3.js is a collection of libraries that enables interaction with an Ethereum Blockchain network. It provides an API to interact with smart contracts, send transactions, and read and write data on the Blockchain. This results in much more efficient management of data and communications within the network. In summary, Web3.js acts as an intermediary between the DApps and the Blockchain.

We have decided to use this library due to the following benefits:

- **Interaction with Smart Contracts:** It allows for the creation and use of smart contract instances.
- **Transaction Management:** It facilitates the sending and receiving of transactions on the Ethereum network.
- **Events and Subscriptions:** It enables applications to subscribe to events and changes on the blockchain.
- **Blockchain Queries:** It provides methods for querying the state and data stored on the blockchain.

Web3js-Quorum is an extension of Web3js designed to interact with Hyperledger Besu blockchain networks. Web3js-Quorum extends the functionalities of Web3js by offering:

- **Compatibility with Private Transactions:** It adds functionality for the creation, sending, and management of private transactions, which is crucial in scenarios where confidentiality is required, such as in enterprise networks.
- **Privacy Group Management:** It allows for the creation and management of privacy groups, ensuring that private transactions are visible only to authorized nodes.
- **Extended JSON-RPC API:** It provides compatibility with the specific extensions of the JSON-RPC API in Hyperledger Besu.

The design of the decentralized application (DApp) would be as shown in Figure 18, with the following key components and relationships:

- **User Interaction:** Each network user interacts with the DApp through a frontend, such as a web browser. This frontend is developed using HTML, CSS, and JavaScript.
- **DApp:** Implemented in Node.js and Express.js. It handles communication with the Blockchain and the smart contract via Web3.js.
- **Smart Contract:** Written in Solidity, the ABI deployed on the Blockchain contains the business logic.

- **Hyperledger Besu Nodes:** Created using Besu technology, there will be a node for each participant in the chain. These nodes represent each user of the application, each with their own access to the frontend. They are responsible for validating transactions and maintaining the private Blockchain.
- **Tessera Nodes:** Created using Tessera technology, these nodes facilitate private transactions and maintain data privacy. They act as an intermediary between the DApp and the Blockchain network, ensuring that sensitive data are visible only to authorized participants.
- **Private Blockchain:** Developed using Ethereum on Hyperledger Besu, this is where the smart contracts are deployed. It is also where transactions are stored and verified.

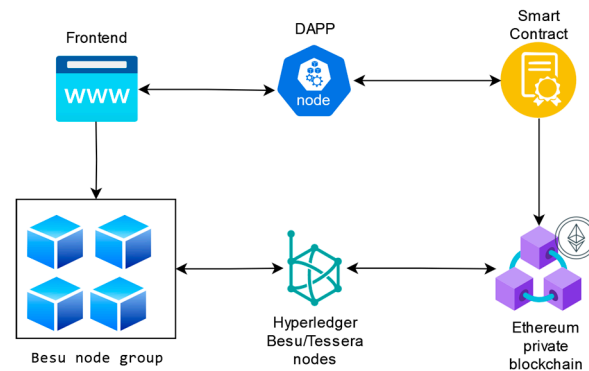


Figure 18. DAPP Design.

5.6. Solution Validation

It is important to highlight that the solution developed and evaluated in this article is a functional prototype, designed to demonstrate the feasibility and benefits of the proposed architecture in a controlled environment.

During the validation phase of the implemented prototype, a series of tests were conducted to confirm the inherent benefits of using Blockchain in multipolicy management. These tests demonstrate the unification of processes, allowing internal operations and interactions between external companies to be carried out more smoothly and coherently. Additionally, the elimination of data duplication in relationships between network participants becomes evident, reinforcing the integrity and consistency of the managed information. Thanks to the implemented model, operational efficiency is optimized, and data consistency across the network is ensured, aligning with the standardization and security objectives that are fundamental to the proposed architecture.

In relation to the standardized policy model, Merkle trees have proven to be an effective tool for managing and expanding the insurance catalog with agility. A practical case was conducted where tests were performed on an initial set of home, auto, and health insurance policies. The inclusion of a new type was proposed: Personal Digital Assets Insurance Policy. The result was a direct integration of the new policy type: it was added as an eligible option in the decentralized user application, and when creating a policy of this new type, a Merkle tree representing the dataset was generated, the hash calculations were performed as previously described, and the data was inserted into the Blockchain network.

The entire process was completed without modifying pre-existing policies and without any disruptions. This process demonstrated the ease with which the architecture allows new types of insurance to be added, ensuring the system's flexibility and scalability.

A preliminary analysis of the performance of the proposed prototype has been conducted. The migration of data to the blockchain system is a process that occurs only once, during the initial deployment, and it is estimated that the migration times are negligible in relation to the volume of data and the required infrastructure. Additionally, the operational capacity of the machines used in production environments must be considered.

Finally, we conducted an empirical analysis of the most frequent operation in this system: data search. From the analysis, we obtained the following results:

With 500 simulated policies, we have the following results:

- Performing a linear search operation requires an average of 250 operations. This is because, under these conditions, approximately half of the elements are reviewed on average. As the data increases, the search time increases linearly.
- Using a binary search through Merkle trees, we found that the search takes $\log_2$ (number of policies), which, applied to the case study of 500 records, results in an average of 9 operations per search. It is important to note that the search time will increase much more slowly as the number of records grows.
- Although the times with the two systems are not significant for the development of operations in the company, on average the use of Merkle trees improves the search by about 50%, having considered the search in all possible positions of the chain.

Although these operations were tested solely using the prototype developed for the proof of concept, we can conclude that this improvement is key to optimizing the system's efficiency, especially as the volume of managed policies increases so in number as in types of policies. For a comprehensive study, including real metrics, it is necessary to consider real-world conditions such as software, hardware, operational time windows, workload, etc.

6. Conclusions

This article has presented a Blockchain-based solution for the management of multi-policies in insurance companies. A standardized policy model has been introduced, which allows for the generalized handling of policies, thereby facilitating business operations in the sector and enhancing commercial relationships between different entities. Additionally, a high degree of interoperability is achieved.

A Merkle tree-based anchoring method within this Blockchain network has been proposed to securely store customer data using smart contracts. This approach enhances key industry objectives, such as data verification and authentication, dispute resolution, and fraud prevention.

The standardization of the proposed policy model offers a flexible and adaptable solution for the adoption of different types of insurance. Through this approach, policies are managed uniformly, facilitating integration and collaboration between various entities within the insurance sector. This capability translates into administrative ease and simplification, enabling the business model to expand into new market niches, ensuring that the proposed solution can quickly adapt to new market demands.

The use of Merkle trees as an insertion mechanism strengthens this standardized model by providing a flexible and efficient method for data management. Under the selected strategy, where each policy is represented by a complete and independent tree, it ensures that any updates or additions of new policies are made without altering existing ones. Insurance companies gain significant flexibility in policy management, along with a highly scalable solution. It also provides a robust verification mechanism for dispute resolution and fraud prevention, allowing for precise identification of changes made to different policies. A complete Blockchain-based architecture has been implemented to support this model, utilizing open technology like Ethereum but within a private network.

This strategic choice not only improves the integrity, authenticity, and availability of data thanks to the inherent features of Blockchain, but it has also allowed us to simplify technical management, eliminate operational redundancies, and enhance privacy by avoiding system exposure.

Overall, these contributions represent a significant step forward in the digitalization and modernization of the insurance sector, with potential impacts that could extend to other fields where data security, efficiency, and integrity are paramount.

Our future work will focus on improving search times by optimizing the anchoring mechanisms within the blockchain network. In particular, we aim to explore the possibility

of using additional data structures as well as other efficient algorithms that have been widely discussed in the literature. These methods could offer advantages not only in the insurance sector but also in other industries, such as transportation and distribution, where document integrity and data security are equally critical. Although blockchain has proven to be a robust technology for ensuring data integrity, our goal is to continue exploring different ways to optimize and customize this technology for various business applications.

Author Contributions: Methodology, A.R. and R.H. Software, A.R. Investigation, A.R. Resources, A.R. Writing—original draft, A.R. Writing—review & editing, R.H. Supervision, R.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The original contributions presented in the study are included in the article, further inquiries can be directed to the corresponding authors.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

In this section, you can find some screenshots of the DAPP prototype made for the proof of concept.

The DAPP has a basic login to access the application, as can be seen in Figure A1:

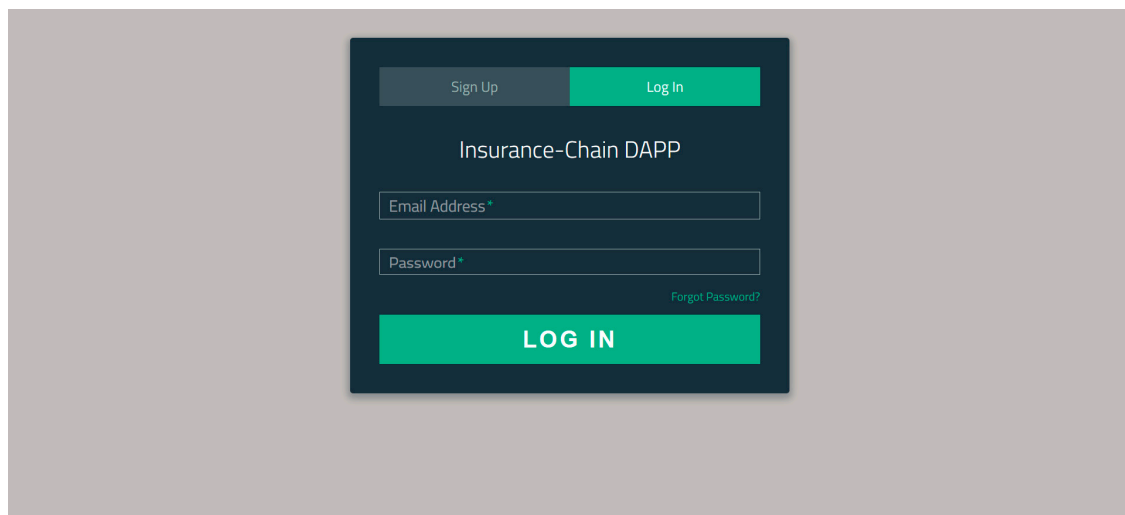


Figure A1. DAPP Login.

Once logged into the application, we will access an initial view of it as we can see in Figure A2. We can find a side navigation menu with the different functions that will be implemented.

If we click on the contract policy action, a new view will load. As can be seen in Figure A3, different steps begin to appear with information that represents the policy modeling detailed above.

In Figure A4, we have another step where, for example, a life insurance policy has been selected and the related information appears. The process continues until all the modeled information is complete.

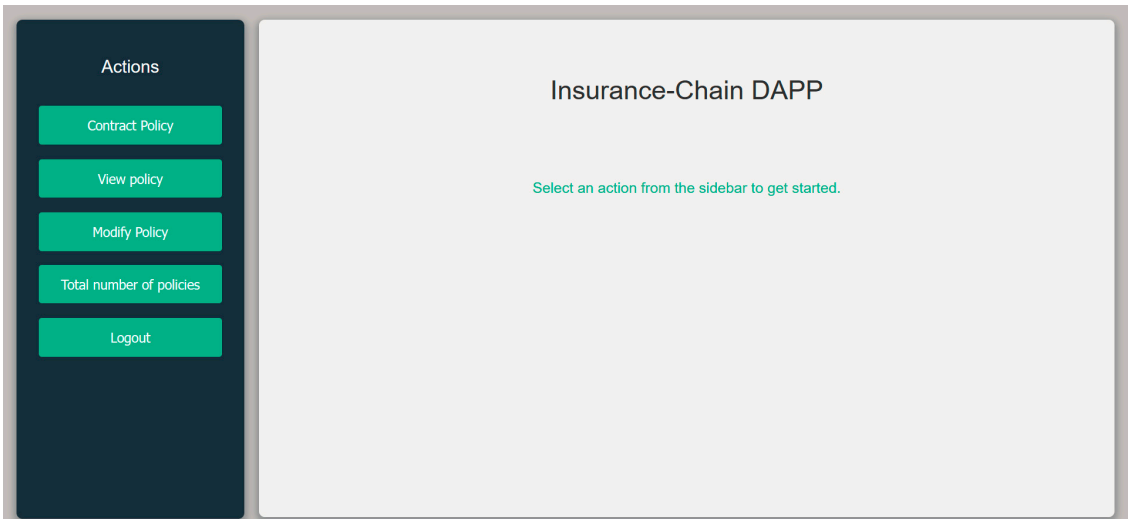


Figure A2. DAPP initial view.

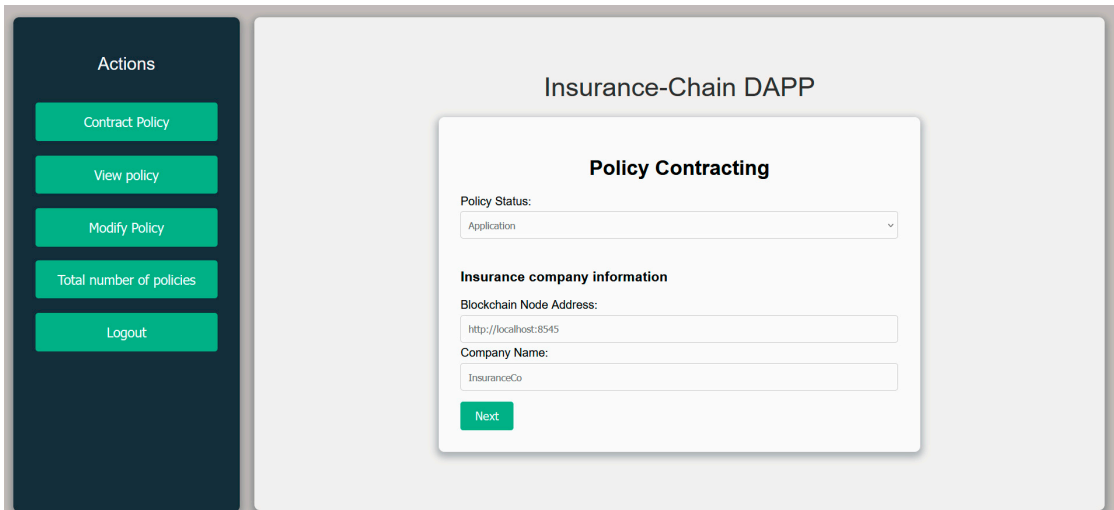


Figure A3. Contract Policy operation.

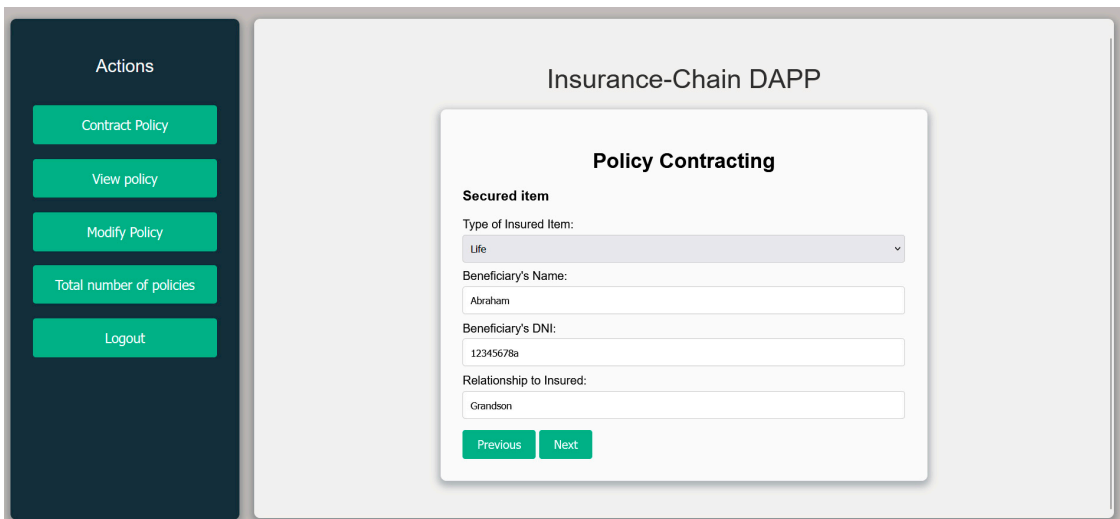


Figure A4. Contract Policy operation successive step.

Once the creation of the policy is completed, if we look at the application logs, we can see how the transaction to the Blockchain is produced, as we see in Figure A5:

[illegible]

Figure A5. Blockchain transaction log.

Let's look at one more example of an operation. The view policy action performs a search on the Blockchain network using the policy number that was generated during creation. The result can be seen in Figure A6:

The screenshot displays the 'Insurance-Chain DAPP' interface. On the left is a dark blue sidebar with white text for 'Actions', containing five green buttons: 'Contract Policy', 'View policy', 'Modify Policy', 'Total number of policies', and 'Logout'. The main area has a light gray background with the title 'Insurance-Chain DAPP' at the top. Below the title is a white card titled 'Search Policy'. Inside this card, there is a search form with the label 'Enter Policy Number:' followed by a text input field containing 'Enter policy number' and a green 'Search' button. Below the search form, the 'Policy Details' section lists four fields: 'Policy Number:' with value '172743266018412345678', 'Holder DNI:' with value '12345678z', 'Merkle Root:' with a long hexadecimal string, and 'Timestamp:' with value '27/9/2024, 12:24:26'.

Figure A6. View policy action.

References

1. Eling, M.; Lehmann, M. The impact of digitalization on the insurance value chain and the insurability of risks. *Geneva Pap. Risk Insur. Issues Pract.* **2018**, *43*, 359–396. [[CrossRef](#)]
2. Albrecher, H.; Bommier, A.; Filipović, D.; Koch-Medina, P.; Loisel, S.; Schmeiser, H. Insurance: Models, digitalization, and data science. *Eur. Actuar. J.* **2019**, *9*, 349–360. [[CrossRef](#)]
3. Cappelletto, A. *Technology and the Insurance Industry*; Palgrave-Macmillan: London, UK, 2018.
4. Njegomir, V.; Demko-Rihter, J.; Bojanic, T. Disruptive Technologies in the Operation of Insurance Industry. *Teh. Vjesn.-Tech. Gaz.* **2021**, *28*, 1797–1805.

5. Gatteschi, V.; Lamberti, F.; Demartini, C.; Pranteda, C.; Santamaría, V. Blockchain and Smart Contracts for Insurance: Is the Technology Mature Enough? *Future Internet* **2018**, *10*, 20. [CrossRef]
6. Trivedi, S. Blockchain Framework for Insurance Industry. *Int. J. Innov. Technol. Manag.* **2023**, *20*, 2350034. [CrossRef]
7. Sheth, A.; Subramanian, H. Blockchain and contract theory: Modeling smart contracts using insurance markets. *Manag. Financ.* **2020**, *46*, 803–814. [CrossRef]
8. Raikwar, M.; Mazumdar, S.; Ruj, S.; Sen Gupta, S.; Chattopadhyay, A.; Lam, K. A blockchain framework for insurance processes. In Proceedings of the 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS), Paris, France, 26–28 February 2018; IEEE: New York, NY, USA, 2018; pp. 1–4.
9. Amponsah Anokye, A.; Adekoya Adebayo, F.; Weyori Benjamin, A. Blockchain in Insurance: Exploratory Analysis of Prospects and Threats. *Int. J. Adv. Comput. Sci. Appl.* **2021**, *12*, 445–466. [CrossRef]
10. Zhou, L.; Wang, L.; Sun, Y. MISTore: A Blockchain-Based Medical Insurance Storage System. *J. Med. Syst.* **2018**, *42*, 149. [CrossRef] [PubMed]
11. Buterin, V. Ethereum White Paper: A Next Generation Smart Contract & Decentralized Application Platform. 2013. Available online: <https://ethereum.org/en/whitepaper/> (accessed on 27 September 2024).
12. Loukil, F.; Boukadi, K.; Hussain, R.; Abed, M. CioSy: A Collaborative Blockchain-Based Insurance System. *Electronics* **2021**, *10*, 1343. [CrossRef]
13. Dutta, H.; Nagesh, S.; Talluri, J.; Bhaumik, P. A Solution to Blockchain Smart Contract Based Parametric Transport and Logistics Insurance. *IEEE Trans. Serv. Comput.* **2023**, *16*, 3155–3167. [CrossRef]
14. Yu, K.; Tan, L.; Shang, X.; Huang, J.J.; Srivastava, G.; Chatterjee, P. Efficient and Privacy-Preserving Medical Research Support Platform Against COVID-19: A Blockchain-Based Approach. *IEEE Consum. Electron. Mag.* **2021**, *10*, 111–120. [CrossRef]
15. Mani, V.; Manickam, P.; Alotaibi, Y.; Alghamdi, S.; Khalaf, O.I. Hyperledger Healthchain: Patient-Centric IPFS-Based Storage of Health Records. *Electronics* **2021**, *10*, 3003. [CrossRef]
16. Bianchini, D.; De Antonellis, V.; Garda, M.; Melchiori, M. Resource-Oriented Approach for Effective Blockchain Integration in Intertwined Supply Chains. In Proceedings of the International Conference on Database and Expert Systems Applications, Naples, Italy, 26–28 August 2024; Springer Nature: Cham, Switzerland, 2024; Volume 14911, pp. 18–33.
17. Singh Yadav, A.; Charles, V.; Kumar Pandey, D.; Gupta, S.; Singh Kushwaha, D. Blockchain-based secure privacy-preserving vehicle accident and insurance registration. *Expert Syst. Appl.* **2023**, *230*, 120651. [CrossRef]
18. Nath, I. Data exchange platform to fight insurance fraud on Blockchain. In Proceedings of the IEEE 16th International Conference on Data Mining Workshops (ICDMW), Barcelona, Spain, 12–15 December 2016; pp. 821–825.
19. Dhieb, N.; Ghazzai, H.; Besbes, H.; Massoud, Y. A secure ai-driven architecture for automated insurance systems: Fraud detection and risk measurement. *IEEE Access* **2020**, *8*, 58546–58558. [CrossRef]
20. Cockburn, A. *Writing Effective Use Cases*; Addison-Wesley Professional: Boston, MA, USA, 2000.
21. Sharma, B.; Sekharan, C.; Zuo, F. Merkle-Tree Based Approach for Ensuring Integrity of Electronic Medical Records. In Proceedings of the 9th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), New York, NY, USA, 8–10 November 2018; pp. 983–987.
22. Zhang, W.; Lu, Q.; Yu, Q.; Li, Z.; Liu, Y.; Lo, S.; Chen, S.; Xu, X.; Zhu, L. Blockchain-Based Federated Learning for Device Failure Detection in Industrial IoT. *IEEE Internet Things J.* **2021**, *8*, 5926–5937. [CrossRef]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.