University of Zurich[UZH]

UZH Blockchain Center

Analysis of Implementing a Smart Contract in Weather Insurance using Chainlink Oracles

Bachelor Thesis

Submitted in partial fulfillment of the requirements for the degree of Bachelor of Science

Author

Aeberhard Elia Andreas

Gmeindhusplatz 4
5223 Riniken

Matriculation Number: **19-925-957**

Email: elia.aeberhard@uzh.ch

Supervisor

Prof. Dr Claudio J. Tessone

Blockchain & Distributed Ledger Technologies

Department of Informatics

University of Zurich

Date of Submission: [Date]

Executive Summary

Write this last. It is an overview of your whole thesis, and is between 200-300words.. . .

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Introduction text ?

1.1 Background

In 2016, global disasters accounted for USD 175 billion in economic losses. USD 54 billion of these economic losses were insured, resulting in uninsured losses of USD 121 billion (Swiss Re Institute 2017). These losses highlight the importance of wheather insurance in providing financial protection for individuals, business and governments. For comparison, the international humanitarian assistance reached USD 28 billion in 2015, making the unsinsured losses of 2016 over 4 times that amount (Initiatives 2016).

1.1.1 Existing Weather Insurance

Traditional weather insurance primarily consisted of crop insurance. The policy in this contract was typically conducted bilateral between an individual or business and the insurance company. If a loss incured on the crop of the individual or business due to weather conditions, an assessment had to be done by the insurance company and the insurance payout was determined based on the specific circumstances (Michler et al. 2022).

Crop insurance has some major problems associated with it. One of these problems is the manual process of analyzing and determining the loss in a monetized amount by an insurance company representative. Other problems include systemic risk, where many insurance holders in the same region are in risk of being affected through weather conditions simultaneously, and asymmetric information, where, for example, insurance holders act more riskily than they normally would because they know they are insured (Makki 2002).

A more modern approach to weather insurance compared to the traditional crop insurance is

weather-based index insurance. The key difference here is that weather-based index insurance relies on a measurable variable (such as a temperature drop below a certain threshold or a specific amount of rainfall). The underlying weather data is provided by a reference weather station. The goal is for the criteria (e.g. the temperature threshold) to reflect the financial loss experienced by the insurance holder, for example the loss of a corn field due to adverse weather conditions (Kajwang 2022).

1.2 Problem Statement

Even though the weather-based index insurance approach poses a significant improvement compared to the traditional weather insurance, there are still a lot of problems associated with it. These problems include high administrative costs, delayed payouts, scalability and lack of trust in the underlying systems and insurance companies. (Skees et al. 2008). These limitations reduce accessability and the range of weather insurance solutions for individuals and business, especially in more developing areas.

To address these problems, this thesis proposes a weather insurance solution based on blockchain technology. Through decentralized, transparent systems and globally available weather data, the solution aims to reduce the administrative costs, enable automatic and instant payouts, improve the scalability and encourage more trust among the insurance holders.

1.3 Objectives

The main objective of this thesis is to analyze possible implementations of a blockchain-based weather insurance solution that addresses the challenges and drawbacks of traditional crop insurance und weather-based index insurance. The specific objectives are as follows:

- Identify and analyze the key challenges of current weather insurance solutions.
- Propose a blockchain-based weather insurance design that utlizes decentralized oracles and globally available weather data.
- Evaluate the proposed solution.
- Compare the blockchain-based solution to traditional crop insurance and weather-based index solutions to analyze the potential and improvements in efficiency, transparency and user trust.

Chapter 2

Literature Review

This chapter dives deeper into the existing challenges and limitations of the current weather insurance models with a focus on weather-based index insurance. It then introduces the concept of using smart contracts in insurance in combination with chainlink oracles and google cloud public datasets. Finally, the chapter examines regulatory and technical challenges associated with implementing a blockchain-based weather solution.

2.1 Key limitations in existing weather insurance

In this section every key limitation of existing weather insurance will have its own dedicated subsection.

2.1.1 Administrative costs

In Traditional crop insurance the administrative costs make up 35% to 40% of the insurance outlays while the remaining portion goes towards other costs such as the insurance payout and the reinsurance costs (Glauber 2004). The majority of these administrative costs consist of loss assessment, monitoring, claims, and underwriting expenses.

In index insurance the administrative costs are significantly lower than in crop insurance because the payouts are based on predefined weather indices rather than assessing individual losses. This index-based insurance model also reduces moral hazards since the payouts are triggered by weather events rather than individual actions. (Kusuma et al. 2018) proposes a weather-based index insurance for rice in Indonesia. It is desgined to be cost-effective by basing the insured amount on the cost of inputs (e.g. seed and fertilizer) rather than covering the individual revenue loss.

In section 2.2 the thesis will discuss how smart contracts can be used to lower administrative costs through automated processes even more than weather-based index insurance.

2.1.2 Systemic risk

A key limitation for existing weather insurance is the systemic risk it poses. (Xu et al. 2010) explains how weather risk is systemic in nature, meaning that weather-related events like droughts or floods often affect entire regions rather than isolated areas.In such an event, a large number of insurance holders would file claims simultaneously, making it difficult for the insurance company to payout all these claims at the same time. It further shows that systemic risk is one of the key reasons why existing weather-based insurance markets have struggled and often require government subsidies, especially in the case of crop insurance.

Based on (Salgueiro and Tarrazon-Rodon 2021), which shows how geographic diversification of the insurance solution reduces the systemic risk, this thesis proposes that a solution based on blockchain technology, which allows for global scalability and diversification, could further reduce systemic risk by creating more decentralized risk pools.

2.1.3 Fraud and manipulation

2.1.4 Lack of transparency and trust

2.2 Smart contracts in Insurance

In 2017 the insurance company AXA launched fizzy, a flight insurance based on smart contract technology. It was one of the earliest examplex of the adaption of blockchain technology in the insurance industry. Fizzy was quite simple. A customer entered his flight details and paid a premium for the insurance. Later Fizzy then used an oracle to check whether the flight has been delayed for more than 2 hours and if so would trigger a payment automatically (Hoffmann 2021).

2.3 Chainlink and Google Cloud Public Datasets

For the prototype this thesis will use Chainlink, a decentralized oracle service, in order for our smart contract to access real-world weather data. The decentralized nature of Chainlink improves the security and reliability of the data-inputs (Beniiche 2020).

Specifically, Global Surface Summary of the Day (GSOD) wlil be used as the weather data source. Over 9000 station provide weather data for the dataset with each station having to report a minimum of 4 observations per day of a set of measured variables including temperature, wind speed, pressure among others. (Environmental Information 2023). The entire dataset is hosted on the Google Cloud Platform (GCP) and is updated daily.

Additionally, the Global Forecast System (GFS) is used which provides weather data prediction globally of up to 16 days. It is a numerical system based on four different models (atmosphere,

ocean, land and sea model) and is hosted on the Google Cloud Platform (GCP) alongside Global Surface Summary of the Day (GSOD) (Environmental Information n.d.).

2.4 Technical and regulatory challenges of Smart Contracts

(Gatteschi et al. 2018) expresses concerns that the technology is still being explored and not yet ready for its benefits to become more evident. One challenge faced by smart contracts in the insurance industry is technical readiness. Oracles are the external data sources that provide real-world information to smart contracts. The integrity of the smart contract is dependent on the reliability of these oracles and the validity of their data.

Another important point to consider are the different kinds of legal problems associated with smart contracts. For one there is no accepted universal definition of a smart contract which leads to difficulties in defining their legal status across different jurisdictions. There are also concerns about consumer protection laws since an automated transaction initiated by a smart contract may violate local regulations if consumers are unable to legally contest or reverse it (Ferreira 2021).

Chapter 3

Methodology

3.1 Research Design

3.2 Data Collection

3.3 Prototype development

Chapter 4

Development of the Prototype

4.1 Requirements

In order to provide the blockchain-based weather insurance, our system must fulfill several key requirements. In this section we describe each category of requirements in a dedicated subsection.

4.1.1 Functional Requirements

- **User Interaction**
  - The system must provide a user interface for purchasing policies and triggering payout eligibility checks.
- **Weather Data Integration**
  - The system must be able to receive and process global weather data from reliable and trusted external sources such as Global Surface Summary of the Day (GSOD) and Global Forecast System (GFS)
  - Both historical and forecast data must be available
- **Smart Contract**
  - The smart contract must allow end users to purchase and terminate weather-based insurance policies based on parameters provided by the end user and the external sources.
  - The smart contract must store the policy terms.
  - The smart contract must be able to perform eligibility checks on existing policies.
  - The smart contract must be able to payout funds for policies that have passed eligibility checks.

- **Oracle Integration**
  - The system must use Chainlink oracles to retrieve and verify weather data from GCP datasets.

4.1.2 Non-Functional Requirements

1. **Security**
   - All the bilateral interactions between the smart contract, Chainlink oracles, GCP and the end user must be secure.
2. **Scalability**
   - The system must be able to scale to handle large amounts of policies and end users.
3. **Transparency**
   - All transactions and insurance claims must be recorded on the blockchain for transparency purposes.

4.1.3 Technical Requirements

1. **Blockchain Platform**
   - The smart contract must be deployed on the Ethereum blockchain
2. **Data retrieval**
   - The system must be able to retreive weather data from GCP.
3. **Chainlink Oracle**
   - The system must integrate with a Chainlink node to facilitate data retrieval from GCP.

4.2 Architecture

In this section we will propose the architecture for our blockchain-based weather insurance system. First we present a high-level overview of the system and then elaborate on the most important components in a dedicated subsection for each component.

4.2.1 General Overview

In fig. 4.1 we present an overview of the general architecture of our proposed system. The center of the architecture is the smart contract deployed on the Ethereum blockchain. It manages all policies and handles the payout process. The end user interacts directly with the smart contract through the use of a decentralized app (see section 4.2.2), where they can request insurance policies and eligibility checks.

The weather data is provided by Global Surface Summary of the Day (GSOD) and Global Forecast System (GFS) respectively. These datasets can be accessed via GCP BigQuery and GCP Storage through the GCP API interface. To bridge the gap between the off-chain data from GCP and the on-chain smart contract, the system uses a Chainlink Oracle (see section 4.2.3). This Chainlink Oracle retrieves the weather data from GCP through its API literface and passes it on to the smart contract on the Ethereum blockchain.

This architecture effectively addresses the technical requirements defined in section 4.1.3 and ensures compatible and reliable communications between each component.

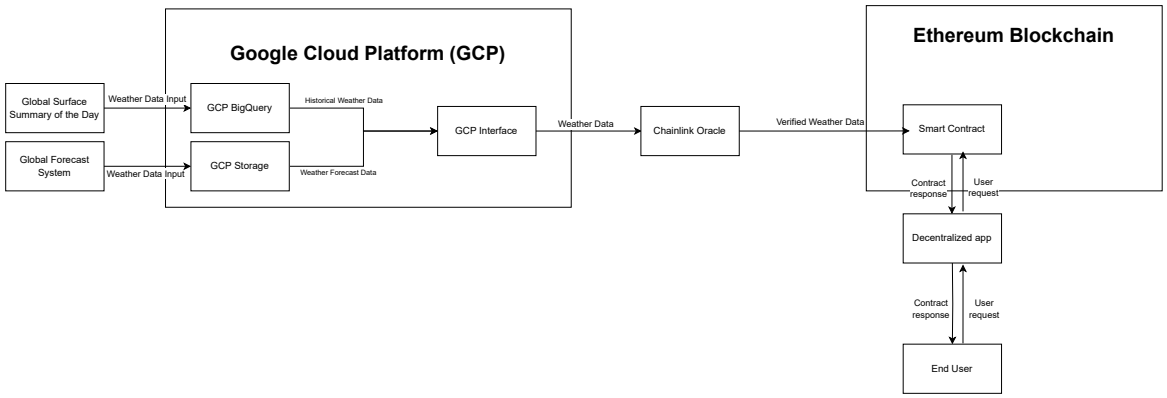

Figure 4.1 – Diagram showing the architecture of the smart contract. *Source: Author's own representation.*

4.2.2 Decentralized App

The decentralized application (DApp) allows a non-technical end user to directly interact with the blockchain. Through its interface a user can request insurance policies, eligibility checks and receive payout funds. Unlike traditional applications, which interact with a centrally managed backend, decentralized applications interact directly with a blockchain.

This interaction requires a digital wallet (for example MetaMask) which enables the user to initiate and sign a transaction when making a request, such as purchasing an insurance policy. In our architecture, the dApp serves as the primary interface in order for the end-users to interact with the smart contract.

4.2.3 Integration of Chainlink Oracles and the Google Cloud Platform

Since the weather data from GSOD and GFS, which is accessed through the GCP datasets, is not directly available from the on-chain environment of the smart contract we need to leverage oracles, which retreive and verify external data before delivering it to the blockchain.

In our proposed system, Chainlink oracles are used to securely interact with the GCP and pass the data on to the smart contract. The Chainlink oracle network consists of globally distributed nodes. When a request is triggered, multiple nodes independently access GCP and receive the weather data specified in the request. If any nodes return results that differ from the majority, they are flagged as potentially malicious. This decentralized approach ensures that only verified and reliable weather data is passed on to the smart contract.

4.3 Data flow

In this section, we examine the two primary data flow scenarios central to our blockchain-based weather insurance system: the process of purchasing a weather insurance policy (see section 4.3.1) and the process of triggering a policy payout (see section 4.3.2). These scenarios represent the fundamental interactions between the end user, the smart contract and the external data sources. In the subsequent chapters we elaborate on the core functionalities utilized in the two scenarios such as the process of retreiving weather data and calculating the policy conditions.

4.3.1 Purchase Weather Insurance Policy

The sequence diagram in fig. 4.2 outlines the steps involved in purchasing a weather insurance policy in our proposed system. The main components are the end user, the smart contract, Chainlink oracle and the Google Cloud Platform (GCP). Each of these components play a central role in enabling the policy purchase and the associated data retrieval and validation processes. Every request between components includes a set of parameters. These parameters contain the necessary information that each component needs in order to perform its role and functions accurately. For example, in a purchase policy request parameters such as location, coverage duration and type of coverage are sent to the smart contract. Table table 4.1 contains all requests with their respective parameters.

The end user starts the process by requesting a weather insurance policy from the smart contract. This interaction happens through the use of a decentralized app (dApp) section 4.2.2, which is not explicitly mentioned in the diagram but can be thought of as the interface enabling the end user to interact directly with the smart contract. Through the dApp the end user can request a weather insurance policy. In table table 4.1 we find the detailed description of each request. In the policy request the user has to specify the conditions of the policy, these include the location, start- and end-date, the type of coverage (for example drought coverage or storm coverage) as well as the

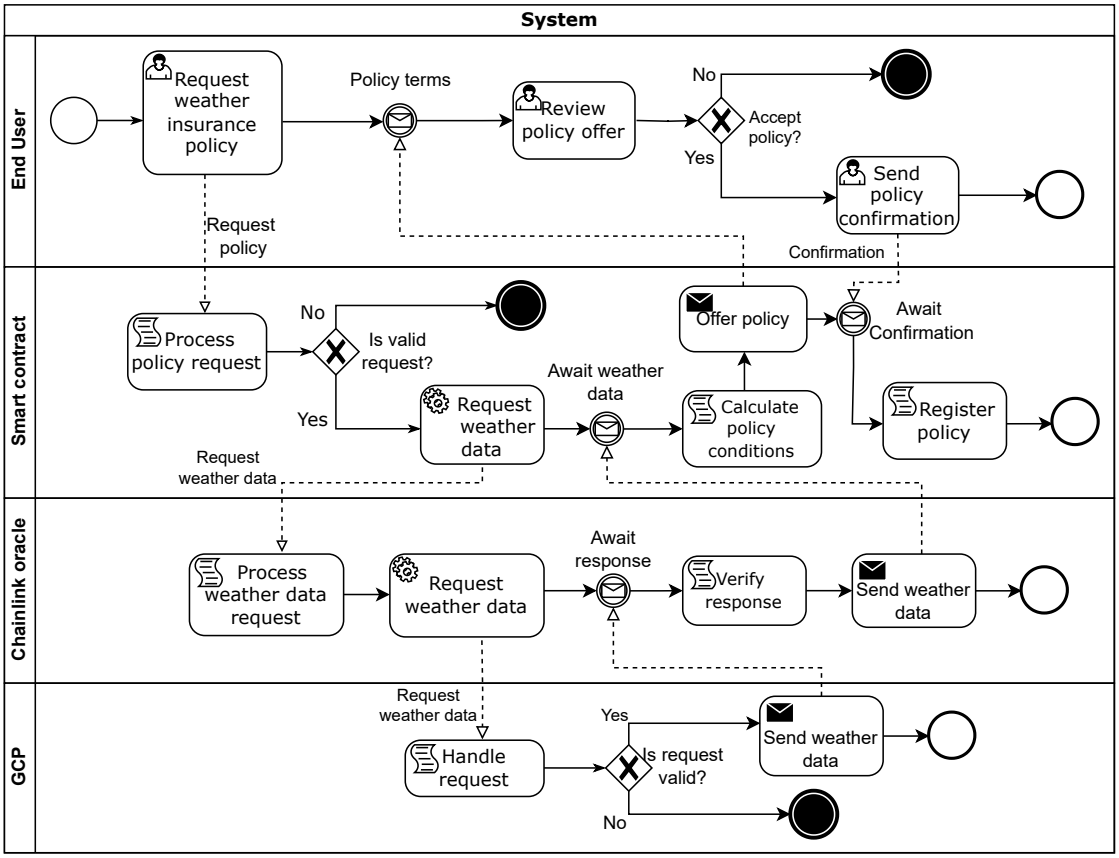

Figure 4.2 – Sequence diagram of purchasing an insurance policy in our proposed system *Source: Author's own representation.*

data type (whether the request should return historical or forecast data).

The smart contract then fetches the necessary weather data needed for the calculation of the policy conditions by sending a request to the Chainlink oracle containing the necessary parameters (see "Weather data request" in table table 4.1). The Chainlink oracle propagates this request to the Google Cloud Platform (GCP). Due to the decentralized nature of Chainlink oracles this request is sent multiple times to the GCP from different Chainlink nodes (see section 4.2.3). The responses from each of these requests are then analyzed and verified by the Chainlink network beofre being sent back to the smart contract. The smart contract calculates the policy conditions (such as the premium amount) and sends an offer to the end user. If the end user accepts the terms offered by the smart contract, then the policy is valid and registered on the blockchain.

Note that in the diagram there are two "Request weather data" requests depicted. One is from the smart contract to the Chainlink oracle and one from the Chainlink oracle to the GCP. Even though these requests differ in their technical details (such as type of request and authorization headers) we have combined them in table table 4.1 as one entry. This was chosen specifically for simplicity

Name	Parameters	Description	Response
Request policy	Location: String Coverage start date: Date Coverage end date: Date Type of coverage: Enum	Requests a policy with the given parameters as the underlying conditions	Binding policy offer including financial conditions and a unique policy ID
Request weather data	Location: String Weather start date: Date Weather end date: Date Type of weather data: Enum Frequency: Enum Data Type: Enum	Requests weather data in a specific date range. Examples for weather data types are rainfall, wind speed and temperature. Frequency indicates the data granularity (e.g. hourly, daily). Data Type can be either forecast or historical.	Weather data in JSON format
Confirmation	Policy ID: String Confirmed: Boolean	Sends a confirmation to the smart contract with the specific policy ID	-

Table 4.1 – Requests in fig. 4.2 with their parameters *Source: Author's own representation.*

purposes, since the diagram is an abstraction of a policy purchase process and leaves space for flexibility in technical implementation.

4.3.2 Trigger Policy Payout

The second data flow is shown in fig. 4.3. It presents the process of triggering a policy payout. This payout happens when the specified policy conditions from section 4.3.1 are fulfilled (assuming a policy has been agreed upon between the end user and the smart contract beforehand).

A key decision in the design of the payout process is determining is what triggers an eligibility check and the following payout process. A possible solution would be to use scheduled tasks that periodically trigger an eligibility request on every policy held by the smart contract. Since the Ethereum blockchain does not support scheduled or automated tasks natively, such a solution would require an external solution. For example Chainlink offers an automation service (Chainlink 2024) with which it would be possible to trigger a scheduled eligibility check on the smart contract (for example once a day). This service would however cost gas which has to be paid by the smart contract and result in a lot of unnecessary transactions. In our proposed solution, the end user initiates the eligibility check instead. This decision prioritizes simplicity and cost-efficiency, avoiding unnecessary transactions that would otherwise increase the policy premium for the end user.

After the end user has initiated the eligibility check through the use of a decentralized App (dApp), the smart contract gathers the weather data for the specific location and covered duration of the

policy. This process of retrieving the relevant weather data is similar to the one in section 4.3.1, with a key distinciton being that the weather data contains past data and not forecast data. Once the weather data is received, the smart contract evaluates whether the conditions for a payout are met and communicates the outcome to the end user. If the end user is eligible for a payout, they can submit a request to the smart contract, which will then transfer the agreed-upon funds as specified in the policy.

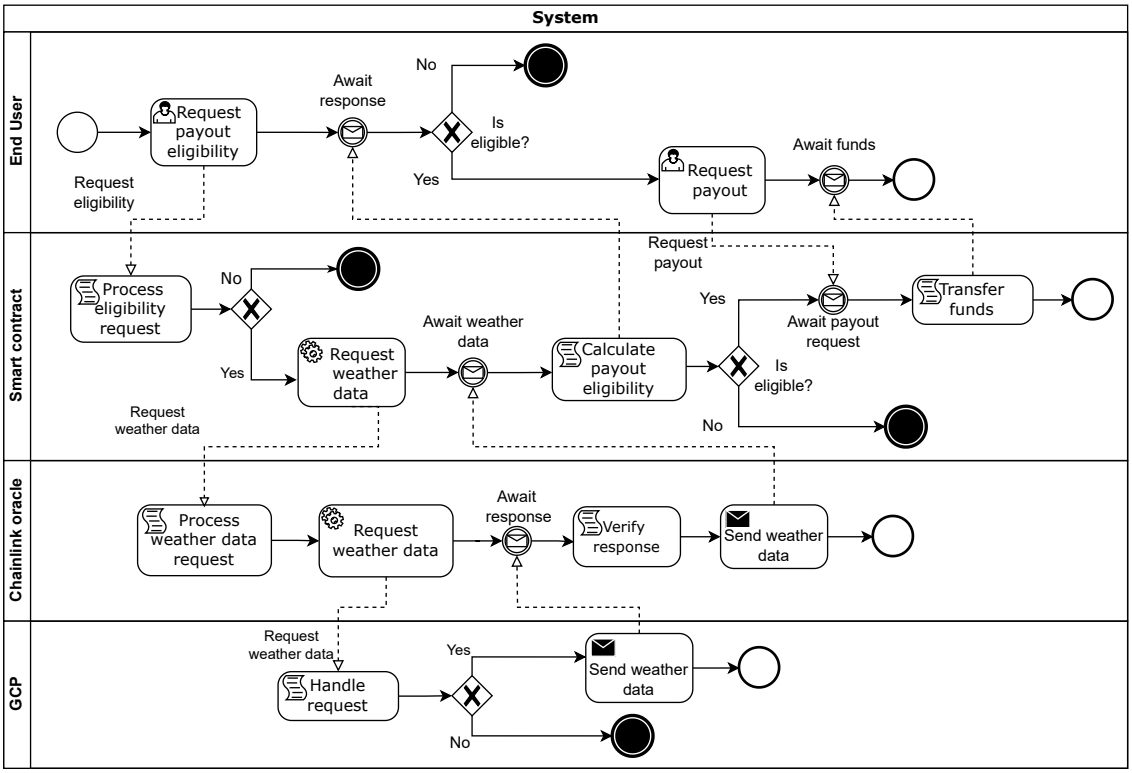

Figure 4.3 – Sequence diagram of triggering a policy payout *Source: Author's own representation.*

4.3.3 Interaction with the Smart Contract

As mentioned in section 4.3.1 and section 4.3.2, the end user communicates with the smart contract through a decentralized app (see section 4.2.2). The dApp interface is indistinguishable from a regular app interface, with the difference being that it interacts directly with the blockchain instead of a traditional backend. Within the dApp the end user can create the requests and enter the relevant parameters. Once the user submits a request, the dApp creates a transaction that is signed using the end user's digital wallet. This transaction is then broadcasted to the blockchain, where it can interact directly with the smart contract.

As the focus of this prototype is primarily on the general architecture of a blockchain-based weather

Name	Parameters	Description	Response
Request eligibility	Policy ID: String	Requests an eligibility check which is then performed by the Smart Contract	Whether the policy is eligible for a payout or not in JSON format
Request weather data	Location: String Weather start date: Date Weather end date: Date Type of weather data: Enum Frequency: Enum Data Type: Enum	Requests weather data in a specific date range. Examples for weather data types are rainfall, wind speed and temperature. Frequency indicates the data granularity (e.g. hourly, daily). Data Type can be either forecast or historical.	Weather data in JSON format
Request payout	Policy ID: String	Requests the payout of funds	Amount of funds specified in the policy

Table 4.2 – Requests fig. 4.3 with their respective parameters *Source: Author's own representation.*

insurance solution and its smart contract functionality, we will not delve further into the specifics of decentralized app (dApp) development, since it is not necessary for the subsequent analysis.

4.3.4 Calculating Policy Conditions

One of the core functions of the smart contract is calculating the policy conditions. In fig. 4.2 it is presented as an internal process of the smart contract. This process involves assessing the parameters provided by the end user in the ”Request policy” request (see table 4.1) as well as the weather data that the smart contract receives from GCP through the ”Request weather data” request (see table 4.1).

In a first validation step, the smart contract ensures that all the parameters are valid, for example that the location is within a supported region and that the coverage is within the allowed limits. Latter is defined through the furtherst available forecast data, which in our solution is 16 days (Environmental Prediction (NCEP) 2016). Next follows the calculation for the policy conditions based on these parameters. These conditions are the following:

- Premium amount (the cost of the policy that is paid by the end user)
- Trigger conditions (specific thresholds or events that must occur for a payout to be issued, for example rainfall above a certain level)

- Coverage limit (the maximum payout amount available under the policy in the case where the conditions for a payout are met)

Note that the coverage limit is defined by the end used in the ”Request policy” request (see table 4.1) but still has to be validated and included in the policy by the smart contract since there may be limits for a maximum coverage amount due to the total funds held by the smart contract.

Calculating the premium amount is the most critical part in the policy conditions. It defines the balance between risk and reward for both the end user and the smart contract. The premium is determined using a formula that considers the following key factors:

- Coverage duration
- Location
- Type of coverage
- Forecast weather data
- The contract margin

This formula represents a fundamental approach for premium calculation. However, it can be further refined by including historical data and actuarial models, allowing for a more dynamic calculation based on real-world risk. Developing a fully-realized mathematical formula suitable for the usage in a production environment is beyond the scope of this thesis. Instead, we focus on identifying and outlining the key factors that are included in the formula. In the analysis chapter (include reference), these factors will be evaluated to compare the premium calculation approach in a blockchain-based insurance system with that of a traditional weather insurance model.

4.3.5 Smart contract fetching weather data

Both in fig. 4.2 and in fig. 4.3 the retrieval of weather data from GCP plays a central role. The security and transparency of our blockchain-based system is only as strong as each component involved in the data retrieval chain. By using Chainlink oracle (include reference), the smart contract can maintain its decentralized nature.

Chapter 5

Analysis and Discussion

5.1 Technological and regulatory barriers of the prototype

5.2 Real-world application of the prototype

5.3 Analysis of smart contracts in the insurance industry

Chapter 6

Summary and Conclusion

6.1 Summary of findings

6.2 Conclusions

6.3 Future work

Appendices

Appendix A

Appendix title 1

Test appendix 1

Bibliography

Beniiche, Abdeljalil (2020). “A study of blockchain oracles”. In: *arXiv preprint arXiv:2004.07140*.

Chainlink (2024). *Chainlink Automation*. Accessed: 2024-11-05.

Environmental Information, NOAA National Centers for (2023). *Global Surface Summary of the Day - GSOD*. Accessed: 2024-10-21.

— (n.d.). *Global Forecast System (GFS)*. Accessed: 2024-10-21.

Environmental Prediction (NCEP), National Centers for (2016). *Global Forecast System (GFS) - Global Spectral Model (GSM) V13.0.2*. https://vlab.noaa.gov/web/gfs/documentation. Accessed: 2024-11-06.

Ferreira, Agata (2021). “Regulating smart contracts: Legal revolution or simply evolution?” In: *Telecommunications Policy* 45.2, p. 102081.

Gatteschi, Valentina, Fabrizio Lamberti, Claudio Demartini, Chiara Pranteda, and Víctor Santamaría (2018). “Blockchain and smart contracts for insurance: Is the technology mature enough?” In: *Future internet* 10.2, p. 20.

Glauber, Joseph W (2004). “Crop insurance reconsidered”. In: *American Journal of Agricultural Economics* 86.5, pp. 1179–1195.

Hoffmann, Christian Hugo (2021). “A double design-science perspective of entrepreneurship–the example of smart contracts in the insurance market”. In: *Journal of Work-Applied Management* 13.1, pp. 69–87.

Initiatives, Development (2016). *Global Humanitarian Assistance Report 2016*. Accessed: October 1, 2024. Development Initiatives.

Kajwang, Ben (2022). “Weather based index insurance and its role in agricultural production”. In: *International Journal of Agriculture* 7.1, pp. 13–25.

Kusuma, Aditya, Bethanna Jackson, and Ilan Noy (2018). “A viable and cost-effective weather index insurance for rice in Indonesia”. In: *The Geneva Risk and Insurance Review* 43, pp. 186–218.

Makki, Shiva (2002). *Crop insurance: inherent problems and innovative solutions*. Ames: Iowa State University Press.

Michler, Jeffrey D, Frederi G Viens, and Gerald E Shively (2022). “Risk, crop yields, and weather index insurance in village India”. In: *Journal of the Agricultural and Applied Economics Association* 1.1, pp. 61–81.

Salgueiro, Andrea Martinez and Maria-Antonia Tarrazon-Rodon (2021). “Is diversification effective in reducing the systemic risk implied by a market for weather index-based insurance in Spain?” In: *International Journal of Disaster Risk Reduction* 62, p. 102345.

Skees, Jerry R et al. (2008). “Challenges for use of index-based weather insurance in lower income countries”. In: *Agricultural Finance Review* 68.1, p. 197.

Swiss Re Institute (2017). *Natural catastrophes and man-made disasters in 2016*. Sigma No. 2/2017. Accessed: October 1, 2024. Swiss Re.

Xu, Wei, Guenther Filler, Martin Odening, and Ostap Okhrin (2010). “On the systemic nature of weather risk”. In: *Agricultural Finance Review* 70.2, pp. 267–284.

Eidesstattliche Erklärung

Der/Die Verfasser/in erklärt an Eides statt, dass er/sie die vorliegende Arbeit selbständig, ohne fremde Hilfe und ohne Benutzung anderer als die angegebenen Hilfsmittel angefertigt hat. Die aus fremden Quellen (einschliesslich elektronischer Quellen) direkt oder indirekt übernommenen Gedanken sind ausnahmslos als solche kenntlich gemacht. Die Arbeit ist in gleicher oder ähnlicher Form oder auszugsweise im Rahmen einer anderen Prüfung noch nicht vorgelegt worden.

.....	.....
Ort, Datum	Unterschrift des/der Verfassers/in