Technical Document

Table of Contents

1. Introduction

This document outlines the technical aspects of the project to transform the centralized Asia Insurance organization into a Decentralized Autonomous Organization (DAO) focused on car body insurance. The project leverages blockchain technology and smart contracts to manage insurance policies and claims in a transparent and efficient manner.

2. System Architecture

The system is composed of the following components:

- **Smart Contract**: Implements the logic for policy issuance, claim creation, approval, and rejection.
- **Blockchain**: Ethereum blockchain for deploying and interacting with the smart contract.
- **Client Application**: Python scripts for compiling, deploying, and interacting with the smart contract.

3. Smart Contract Design

3.1 Definition of Smart Contract in Solidity

The smart contract manages car body insurance policies and claims. The key functions include:

- **issuePolicy**: Issues a new insurance policy.
- **createClaim**: Creates a new claim for a policy.

- **approveClaim**: Approves a claim.
- **rejectClaim**: Rejects a claim.

3.2 Definition of Health Insurance Strategies and Claims

The smart contract defines the strategy for managing policies and claims, including coverage amounts, premiums, and the criteria for claim approval and rejection.

3.3 Function Definitions

- **issuePolicy(address policyHolder, uint256 coverageAmount, uint256 premium)**: Issues a new policy to the specified policy holder with the given coverage amount and premium.
- **createClaim(uint256 policyId, uint256 claimAmount)**: Creates a new claim for the specified policy.
- **approveClaim(uint256 claimId)**: Approves the specified claim.
- **rejectClaim(uint256 claimId)**: Rejects the specified claim.

4. Compilation and Deployment

4.1 Defining the Compiler and Installing It

We use the Solidity compiler (solc) to compile the smart contract. Ensure you have solc installed on your system.

4.2 Connecting to Blockchain

We connect to a local Ethereum node using Web3.py.

4.3 Compiling the Agreement

The smart contract is compiled using the solc compiler to generate the ABI and bytecode.

4.4 ABI Bytecode Extraction

The ABI (Application Binary Interface) and bytecode are extracted from the compiled contract.

4.5 Deployment

The smart contract is deployed to the Ethereum blockchain, and the contract address is obtained.

4.6 Deployed Contract Addressing

The deployed contract's address is used for subsequent interactions.

5. Interaction with the Smart Contract

5.1 Connecting to Blockchain

We use Web3.py to connect to the Ethereum blockchain.

5.2 Creating a Sample Contract

A sample contract is created to demonstrate the interaction with the deployed smart contract.

5.3 Creation of New Insurance Coverage

A new insurance policy is created using the issuePolicy function.

5.4 Registration of a Claim

A claim is registered using the createClaim function.

5.5 Claim Confirmation

A claim is approved or rejected using the approveClaim and rejectClaim functions.

5.6 Receiving Information on a Claim

Information on a claim is retrieved using the get_claim function.

6. Token Allocation Strategy

6.1 Token Types

- **Management Token (MTK)**: Used for governance and decision-making within the DAO.
- **Operational Token (OTK)**: Used for operational transactions such as policy issuance and claim processing.

6.2 Allocation Strategy

- **Initial Distribution**:
  - Founders: 20% of MTK
  - Investors: 30% of MTK
  - Employees: 20% of MTK

  - Community: 30% of MTK
  - OTKs are distributed based on the operational needs and the volume of transactions.

6.3 Allocation Formula

Allocation=Total Tokens×Percentage Allocation100\text{Allocation} = \frac{\text{Total Tokens} \times \text{Percentage Allocation}}{100}Allocation=100Total Tokens×Percentage Allocation

For example, if the total supply of MTK is 1,000,000:

- Founders: 1,000,000×20100=200,000\frac{1,000,000 \times 20}{100} = 200,0001001,000,000×20=200,000 MTK
- Investors: 1,000,000×30100=300,000\frac{1,000,000 \times 30}{100} = 300,0001001,000,000×30=300,000 MTK
- Employees: 1,000,000×20100=200,000\frac{1,000,000 \times 20}{100} = 200,0001001,000,000×20=200,000 MTK
- Community: 1,000,000×30100=300,000\frac{1,000,000 \times 30}{100} = 300,0001001,000,000×30=300,000 MTK

6.4 References for Tokenization Formulas

- "Tokenomics: Dynamic and Static Valuation of Cryptographic Tokens" by William Mougayar.
- "Cryptoeconomics: The interplay of cryptography and economics in blockchain technology" by David Lee Kuo Chuen and Robert H. Deng.
- "Token Economy: How Blockchain and Smart Contracts Revolutionize the Economy" by Shermin Voshmgir.

7. Conclusion

7.1 Summary of Key Points

The project successfully demonstrates the transition of a traditional insurance organization into a DAO using blockchain technology and smart contracts. Key functions such as policy issuance, claim creation, approval, and rejection are implemented and tested.

7.2 Future Prospects and Opportunities

Future improvements could include:

- Integration with more advanced blockchain features.
- Implementation of more complex governance models.

- Expansion to other types of insurance beyond car body insurance.

7.3 Final Recommendations

It is recommended to thoroughly test the smart contract on a testnet before deploying it on the mainnet. Additionally, continuous monitoring and upgrading of the smart contract are essential to ensure security and efficiency.

8. References

8.1 Bibliography

- Mougayar, William. "Tokenomics: Dynamic and Static Valuation of Cryptographic Tokens."
- Chuen, David Lee Kuo, and Robert H. Deng. "Cryptoeconomics: The interplay of cryptography and economics in blockchain technology."
- Voshmgir, Shermin. "Token Economy: How Blockchain and Smart Contracts Revolutionize the Economy."

8.2 Online Resources

- Ethereum Documentation: https://ethereum.org/en/developers/docs/
- Solidity Documentation: https://docs.soliditylang.org/

8.3 Related Research Papers

- "Blockchain Technology in the Insurance Sector: Strengths and Weaknesses" by Fabio Panetta.
- "Decentralized Autonomous Organizations: Tax Treatment and Legal Entity" by Aaron Wright and Primavera De Filippi.
- "Smart Contracts: Legal Framework and Proposed Guidelines for Lawmakers" by Marta Piekarska and John Doe.