卒 業 研 究 報 告 書

題　目　Implementation of Employment Contract using Smart Contract

指導教員　水野　修　教授

崔　恩瀞　助教

京都工芸繊維大学 工芸科学部 情報工学課程

学生番号　18122027

氏　名　Zulfakar Ashraf Bin Jamal

令和4年2月10日提出

Implementation of Employment Contractusing Smart Contract

令和 4 年 2 月 10 日 **18122027 Zulfakar Ashraf Bin Jamal**

Abstract

Ethereum is a technology that is home to digital money, global payments, and applications[1]. Ethereum is a blockchain network that has smart contract functionality. The runtime environment for transactions in ethereum is called Ethereum Virtual Machine (EVM). In other words, EVM is a software framework that allows developers to create Ethereum-based decentralized applications (DApps). Due to EVM popularity and its wide adoption from developers, many emerging new blockchain are including EVM in their architecture. Thus, enabling the application built on the ethereum network to be deployed in their network without making any changes. This research aims to find the possibilities of smart contracts for employment contracts as opposed to traditional paper contracts. We will start to design and plan the structure of smart contracts for signing and employment contracts (RQ1), and then find out their benefits and their problems (RQ2). We start with designing the smart contract and deploy it to the Ethereum network making sure we take the necessary approach to guarantee contract safety. Then, we create a web app to make it easier to interact with the contract. It should be noted that smart contracts are immutable once deployed. Hence, it is crucial to test the application to ensure that it can be safely used for the public. Finally, we analyze some data regarding wage theft and current Ethereum loan protocol and find out how these smart contract implementation of Employment contract can solve some wage theft issues.

Index

1. Introduction

Cryptocurrency utilizes cryptographic tools to enable a decentralized currency managed in a distributed ledger. Due to the rules that control what can and cannot be done to edit the ledger, it operates similar to a regular currency. Bitcoin is the first cryptocurrency that has been created. A bitcoin address cannot spend more Bitcoin than it has received. All transactions on Bitcoin and many other blockchains are governed by these rules.

Ethereum also works on similar technology to Bitcoin. While Ethereum has its own native cryptocurrency (Ether), which follows nearly identical intuitive rules as Bitcoin, it also allows for a far better feature. Since its creation, it has gained the attention of cryptocurrency communities due to this feature that are called smart contract. This complex feature works like a distributed state machine rather than a distributed ledger. The state of Ethereum is a massive data structure that contains not only all accounts and balances, but also a machine state that can change from block to block according to a set of rules and run arbitrary machine code [2]. The Ethereum Virtual Machine (EVM) specifies the rules for altering state from one block to the next. EVM allows developers to create Ethereum-based decentralized applications or popularly known as DApps.

EVM wide adoption has caused many new blockchain systems to implement its architecture. Some of the examples of these blockchain projects are Polygon, Avalanche and Fantom. Some of these projects aim to tackle Ethereum's extremely high gas fee due to its scaling problem from mass adoption. Many newly emerging blockchain technologies for the past 2-3 years are made to be EVM compatible, enabling the same code to be deployed in multiple different cryptocurrency networks. Thus, it is becoming more important to innovate and find new use cases for blockchain technology. This research aims at possibilities of smart contracts for employment contracts as opposed to traditional paper contracts.

RQ1 How to implement an employment contract using smart contract? We design smart contract that mimic employment contract in the real world. We will also tackle the smart contract in terms of security.

RQ2 What are the benefits of using smart contract to sign employment contract? Due to blockchain immutability and its decentralized nature, it is perfect for signing a contract. Plus, self-executing contract with the terms of the agreement between employee and employer being directly written into lines of code will enable easier contract execution as the presence of a lawyer is unnecessary.

2. Background

This section summarizes the concepts that are the key to this study.

Blockchain A blockchain is a decentralized, chronological database of transactions that is shared and maintained amongst nodes in a peer-to-peer network.

Bitcoin A decentralized digital currency that can be sent from user to user on the peer-to-peer bitcoin network without the use of intermediaries. It has no central bank or single administrator. Network nodes use cryptography to verify transactions, which are then stored in a public distributed ledger called a blockchain.

Ethereum A decentralized, open-source blockchain with smart contract functionality. Ethereum, in particular, hosts and executes smart contracts. Ethereum is sometimes referred to as a programmable blockchain due to these characteristics.

Smart Contract A computer program or transaction protocol that is designed to automate the execution, control, and documentation of legally significant events and activities in accordance with the conditions of a contract or agreement. The decrease of the need for trusted intermediaries, arbitration and enforcement costs, fraud losses, and malicious and inadvertent exceptions are all goals of smart contracts.

Ethereum Virtual Machine Ethereum Virtual Machine (EVM) specifies the rules for altering state from one block to the next. Ethereum is a distributed state machine rather than a distributed ledger. The state of Ethereum is a massive data structure that contains not only all accounts and balances, but also a machine state that can change from block to block according to a set of rules and run arbitrary machine code.

Ethereum Accounts User accounts and smart contract accounts are the two types of accounts in Ethereum. A unique ID is assigned to both accounts. Smart contract accounts can be deployed by user accounts. The contract's creator is frequently referred to as the deployer. A contract's deployment is similar to object instantiation. A new instance is formed from the smart contract and stored in the blockchain. A smart contract's code cannot be modified or replaced once it has been launched. User accounts can also execute smart contracts that have been deployed by submitting transactions that call the function

or method defined in the contracts.

ERC-20 The technical standard for all smart contracts on the Ethereum blockchain for token implementation. It establishes a set of rules that must be followed by all Ethereum-based fungible tokens.

ERC-721 It is a standard for representing ownership of non-fungible tokens, or tokens that aren't interchangeable. It establishes a set of rules that must be followed by all Ethereum-based non-fungible tokens.

Contract A written or oral agreement that is intended to be legally enforceable, especially one involving employment, sales, or tenancy.

Wage theft Occurs when an employee is denied wages or benefits that are legally owed to them by their employer. Action by employer that can be classified as wage theft include failing to pay overtime, violating minimum wage laws, misclassifying employees as independent contractors, making illegal pay deductions, forcing employees to work outside their work time, failing to pay annual leave or holiday entitlements, or simply failing to pay an employee at all.

3. Application Setup

3.1 Preleminary Study

When designing a smart contract, it is very important to consider its security. Since the early days of ethereum, many smart contracts have been exploited. One of the biggest exploits happened in June 2016 where almost 3.6 million Ether worth of ethereum were stolen and this situation ends up with hard forking the ethereum network to revert the transaction and fix the vulnerability [3]. While this was a viable option at that time, it is much harder to execute nowadays as the majority of the network miner needs to agree to perform a hard fork. Before designing the smart contract this research, we are going to consider Rentrancy and Integer Overflow.

3.1.1 Reentrancy Attack

Reentrancy refers to the execution of a function again after it is interrupted in the middle. While the behavior itself might not be an issue, the smart contract can be exploited if not enough thought were put into the function logic. The Dao Hack were cause by this in which the withdraw function were called multiple times by the hacker causing the ethereum funds inside the contract to be drained. However, with the release of Solidity v0.8.0 reentrancy problems have been addressed.

3.1.2 Integer Overflow and Underflow

When a number is incremented above its maximum value Overflow occurs. In contrast, underflow happens when a number is decremented well below its minimum value. It is really important to consider this aspect when designing our contract. Especially when making a certain withdrawal or deposit function in a smart contract. Previously, it was common to use SafeMath Library by OpenZeppelin when performing any number calculation inside smart contracts. Interestingly, this problem was also addressed in Solidity v0.8.0. By default, execution of a transaction automatically fails and reverts if an overflow or underflow occurs during a transaction.

3.2 Protocol Setup

Figure 3.1 represents the overall structure of smart contracts for employment contracts in this research. Below, we are going to discuss the details of each of the smart contract components.

3.2.1 YenToken

YenToken is a representation of Japanese Yen stablecoin using ERC-20 token implementation. While it is possible to choose ethereum tokens to pay worker salary, its heavy price fluctuation might cause some problems making it hard to be accepted as a currency for the time being. In this research, using stablecoin is a better approach as it is able to represent current employment contract better.

3.2.2 Salary

SalaryContract is the core contract for this research. The main logic for employment contracts are located in this contract. It holds the agreement between the employer and the employee. This contract has a total of 6 parameters. Table 3.1 refers to each parameter and its corresponding description.

To use the contract, the employer must first fund the contract. The amount that needs to be funded is the total amount of token that the employee will receive by the end of the contract. After funding, the employee can then sign the contract. In this stage, a key to retrieve the contract funds in the form of ERC-721 token or commonly known as NFT will be sent to the employee wallet address. When redeeming salary in the future, it is necessary to hold the nft key corresponding to the contract. It is also to be noted that in this case, anyone who owns the key is able to redeem the salary inside the contract. The redeemable salary are calculated based on the time since the start of the contract and employee daily salary

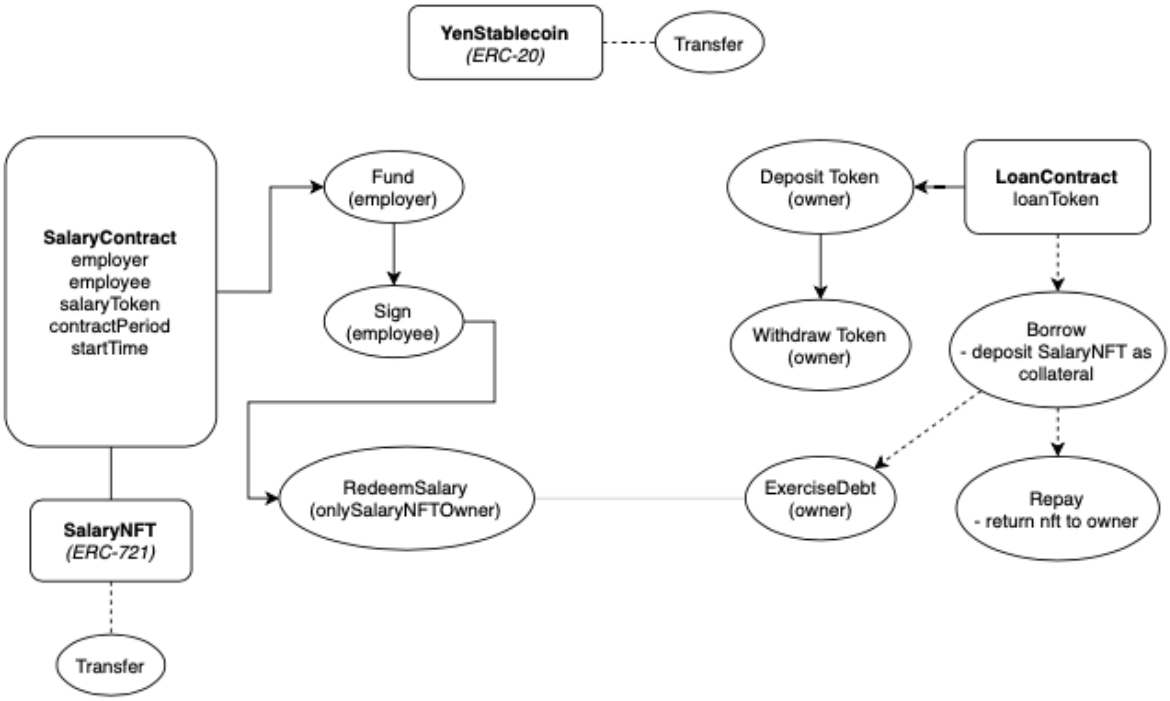

Figure 3.1 Overall structure of Employment Contract Implementation

Table 3.1 Salary contract deploy parameter and description

Parameter	Description
employer	Wallet address of employer
employee	Wallet address of employee
tokenAddress	ERC-20 token that are going to be the currency for paying the salary and loan. It is an imaginary Yen stablecoin
dailySalary	Daily salary of employee
contractPeriod	Total contract period of the contract in days
startTime	Start time of the contract

3.2.3 Loan

Loan contract is actually created to enable borrowing of money using the NFT key of the Salary contract as collateral. As the salary contract only allows employees to withdraw money after a certain amount of time, employees can use the key to borrow money from this loan contract for a certain amount of interest. This is beneficial for the borrower as they can borrow money ahead of time, and the lender as they will gain the interest from lending the money. The lender's money is also protected in this case since the borrower can only borrow a portion of available funds in the smart contract. In case the borrower cannot return the borrowed money, the lender can use the nft key to redeem the borrower salary from the salary contract.

To use the contract, the loan contract owner must first deposit the yenToken into the loan contract. It is also possible to withdraw the token. Owner of the contract can specify the amount of the yenToken they want to deposit or withdraw. Then, someone who owns an NFT key for a salary contract can then borrow money from the loan contract. Using the nft key as collateral, they can borrow up to half the amount of available funds in the corresponding salary contract. There is two possible outcome after borrowing:

(1) Borrower repay back the yenToken that they had borrowed with a 10 percent interest.
(2) Borrower could not pay back the yenToken that they had borrowed and loan contract owner used the nft key to redeem borrower salary.

In both of these cases, the loan contract owners are able to make money.

3.3 Deployment and Testing

3.3.1 Deployment order

Figure 3.2 shows the deployment flow of the employment contract. In this research, the contract were deployed in a locally created ethereum network for easier testing. In a local ethereum network, the time for a transaction to be approved and mined is almost instantaneous. Referring to the figure, it is necessary to deploy the yenToken contract

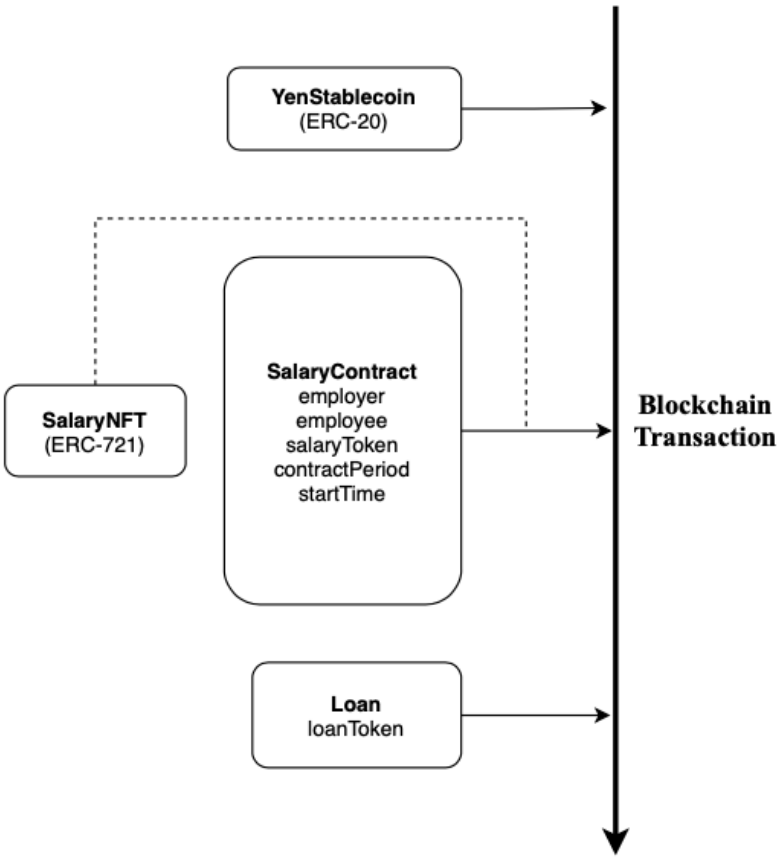

Figure 3.2 Overall structure of Contract Implementation

first as it is the primary token that is going to be used as currency exchange in Salary and Loan contract. SalaryNFT contract are created the moment SalaryContract are deployed. Loan contract are deployed last as the loaning protocol depends on work with the SalaryContract.

3.3.2 Enabling contract usage

While technically it is possible to use a smart contract by directly executing its method from the blockchain, creating a front end page is important to enable wide adoption. In this section, we had created a front end web page to track token ownership, create salary contract and employment contract. The web app is comprised of the following three pages.

(1) **Token Ownership**

Figure 3.3 refers to the page created that are used to track the owner of Yen Stablecoin and the owner of NFT. When testing the salary contract and loan contract, it is extremely helpful to know who owns how much stablecoin token at a certain time. Also, since the contract or wallet that owns the nft key is able to redeem salary in the corresponding salary contract, it is important to see who owns them.

(2) **Salary contract Deployment and tracking**

Figure 3.4 refers to the page created that enables users to deploy a salary contract. As mentioned in the previous section, users need to provide the parameter for creating the smart contract. Once a salary contract is deployed, it can then be tracked. So, it is easier to get detailed information regarding a salary contract. It should be noted when funding yenToken to the salary contract, a transaction needs to be made to allow the contract to spend the yenToken. This is in accordance with the ERC-20 token guideline made by OpenZeppelin.

(3) **Loan contract deployment and tracking**

Figure 3.5 refers to the page created that is used to deploy a loan contract. The Deployer needs to provide the yenToken address for creating the loan contract. Once a loan contract is deployed, it can then be tracked. So, it is easier to get detailed

information regarding a salary contract. Plus all the methods can then be executed from the page. Referring to the image, buttons coloured in blue (Deposit, Withdraw and Exercise Debt) can only be executed by the loan contract deployer while buttons in green (Borrow and Repay) can only be executed by the borrower. It should be noted when depositing yenToken to the loan contract, a transaction needs to be made to allow the contract to spend the yen token. This is in accordance with the ERC-20 token guideline made by OpenZeppelin.

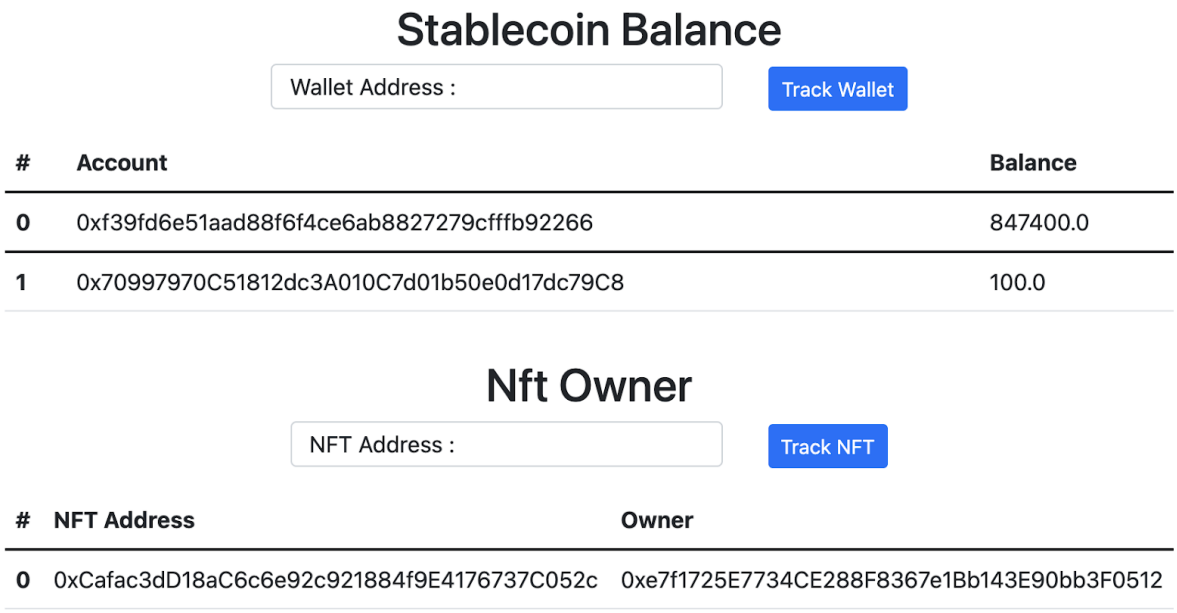

Figure 3.3 Webpage for tracking token ownership

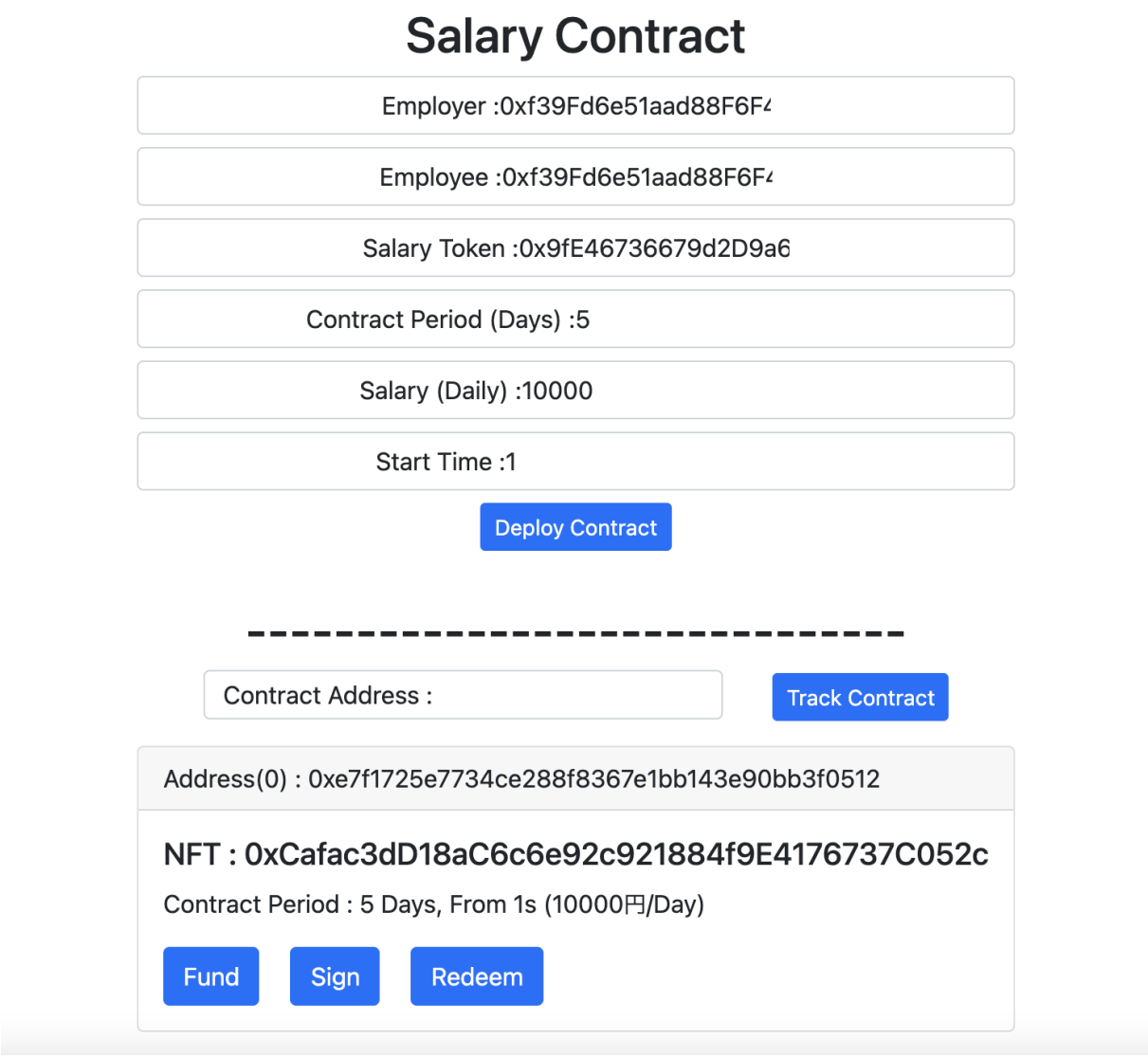

Figure 3.4 Webpage for Salary contract deployment and tracking

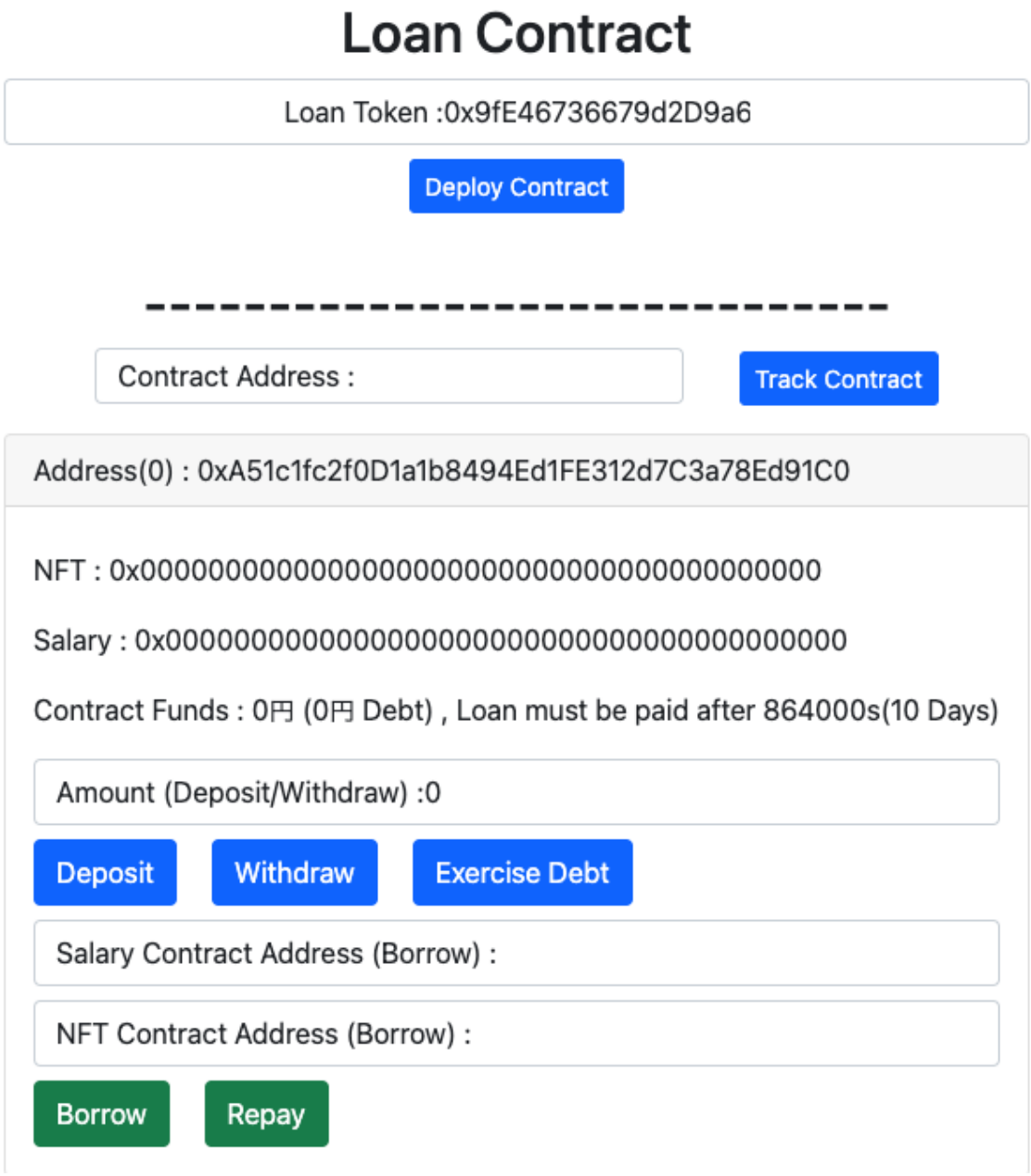

Figure 3.5 Webpage for Loan contract deployment and tracking

4. Empirical Study

4.1 RQ1 How to implement an employment contract using a smart contract?

Motivation The key motivation in this research is to create employment contracts using smart contract as it can be a better alternative to traditional employment contracts. Employment contract are sometimes hard to understand. When a contract's terms or rules were broken by any of the parties, it might be necessary to get a lawyer to get the proper judgment. This process is costly and highly inefficient. Using smart contracts might be a solution to this problem as the contract term and rules are going to be executed indefinitely. Users can enforce their right as long as it is within the rule of the smart contract. In the previous section, we have discussed smart contract and their common vulnerabilities.

Approach To implement an employment contract using a smart contract, we have followed the procedure below.

(1) Designing the smart contract
(2) Deploy the smart contract in ethereum network
(3) Create web app to interact with contract

In step 1, we design the smart contract. In real world cases, salary contract deal with real money, so security should be the top priority. Next, the funds for the salary are going to be deposited in the contract itself. For the employer to withdraw funds there are multiple ways to enable withdrawing salary. Below are different ways to withdraw salary.

Method 1) Allow withdrawal only by employee address.
Method 2) Implement a key system.

We can implement Method 1 by checking inside the salary withdrawal function to ensure that only employees can withdraw the salary. This method is fairly simple and easy to implement but we can make the contract more flexible using the next method.

To implement a key system in Method 2, we can choose to make an NFT as the key to the Salary Contract. NFTs are growing rapidly in the world of digital art and collectibles. However, digital art is merely one application of NFTs. They can be used to indicate ownership of any one-of-a-kind asset, such as a deed for a digital or physical item[4]. It could be thought that the NFT are pegged to the salary fund contained inside the salary contract.

In step 2, the contract needs to be deployed somewhere so that it is accessible by people after it is created. We choose to deploy it in a local ethereum network. The Ethereum development environment used is Hardhat.

Finally in step 3, To enable the people to interact with the contract easily, we had also created a web app using React Framework.

Result In step 1, for designing the smart contract, we made sure to program it using Solidity 0.8.0 to ensure contract security. By doing so, we can guarantee to a certain degree that our contract are at least free from Integer Overflow and Reentrancy hack. We also found out by implementing a key system, many possible new applications are possible from doing so. While the initial goal for this research is to implement an employment contract using smart contract, we had also created a loan protocol using the salary contract. For step 2, we had deployed the contract to the local ethereum network using Hardhat, an ethereum development environment. For this research, using a local ethereum network is suitable as transactions can be approved and mined quickly. Finally, a web app is created to enable interaction with the contract. Web app is also important because it can be used for testing the logic of deployed smart contract.

In regards to how the implement an employment contract using smart contract, we have created a system in which a contract between employee and employee can be made through smart contract. Once deployed, we can be sure that the contract cannot be modified anymore due to smart contract immutability. In future research, we can use the key architectural pattern used in this smart contract implementation as a starting point.

4.2 RQ2 What are the benefits of using smart contract to sign employment contract?

Motivation In the first research question, we have discussed the way to implement employment contract using smart contract. But we have not discussed about the benefit of using smart contract as opposed to traditional employment contract Determining the benefit of implementing employment contract using smart contract is really important so that we can compare the system implemented in this research to the current employment contract.

Approach In this RQ, We analyzed the topic of wage theft. Type of Wage theft includes misclassification and overtime, minimum wage violation and not receiving last paycheck[5]. Misclassification happens when employees were not compensated for their overtime even though they should be receiving salary. Minimum wage violations occur when the workers compensation is less than the minimum salary. Finally, employees not receiving their last paycheck violation occur when workers are denied their last month salary before quitting. When these kinds of wage theft occur, employees can file a report to their local department of labor[6]. When reporting wage theft, document that need to be prepared includes:

- Name and contact information of employee
- The name of the company employee is working
- Location of the company
- Phone number of the company
- Manager or company owner's name
- Type of work and method of salary payment

Next we also analyzed the current existing loan protocol. One of the earliest and the most popular lending and borrowing protocol in ethereum network is called AAVE. Aave is an open source and non-custodial liquidity protocol for earning interest on deposits and borrowing assets [7].

Figure 4.1 shows the webpage of AAVE protocol. Lenders must deposit funds into liquidity pools in order to transact on Aave, and consumers may then borrow from these

Assets	Market size	Total borrowed	Deposit APY	Variable Borrow APY	Stable Borrow APY
Binance USD	32.4M	9.39M	0.38 % 0.09 % APR	1.46 % 0.61 % APR	–
DAI	1.33B	1B	2.61 % 0.51 % APR	3.84 % 1.36 % APR	12.62 %
Gemini Dollar	13.36M	6.56M	1.09 %	2.49 %	–
sUSD	33.13M	22.03M	1.78 % 0.60 % APR	3.38 % 1.80 % APR	–
TrueUSD	99.07M	60.08M	1.72 % 0.28 % APR	3.08 % 0.94 % APR	12.21 %
USD Coin	3B	2.1B	2.01 % 0.50 % APR	3.15 % 1.44 % APR	11.13 %
USDP	14.37M	7.96M	1.24 % 0.38 % APR	2.49 % 1.38 % APR	–
USDT Coin	1.02B	733.31M	2.15 % 0.42 % APR	3.24 % 1.20 % APR	12.29 %
Balancer	567.49K	160.41K	1.00 % 0.91 % APR	4.49 %	–
Ethereum	1.34M	39.19K	0.01 % 0.28 % APR	0.36 %	3.51 %

Figure 4.1 Webpage for AAVE protocol

pools. To protect against volatility, each pool sets aside assets as reserves. These reserves also assist lenders get their money back when they're ready to leave the protocol. For example, users can provide Ethereum token and borrow USDC - a US dollar backed stablecoin, that corresponds to the amount of ethereum that they had deposited. Nearly 20 different cryptocurrencies are available for lending and borrowing on Aave. DAI, ETH, BAT, LINK, MANA, MKR, SNX, USDT, USDC, TUSD, USDT, sUSD, BUSD, KNC, LINK, wBTC, ZRX, and, of course, LEND are among the coins in this stockpile. However, not all cryptos can be used as collateral.

Result The benefit of using Smart contract implementation of Employment contract are as follow:

(1) Preventing wage theft

Type of Wage theft includes misclassification and overtime, minimum wage violation and not receiving last paycheck. Tackling all this problem highly depends on how the smart contract logic is implemented. The employment contracts in this research are able to prevent employees from not receiving their last paycheck when they quit. While the smart contract implemented in this research does not solve all the critical factors of wage theft, it aims to be a suggestion and guideline on how employment contracts can be implemented using smart contracts moving forward. Referring to the document necessary for reporting wage theft and considering the effort necessary to exercise an employment contract if any of the parties breach contract clause, using smart contract might be the solution.

(2) Allow easier borrowing using contract keys as collateral.

With the way we design the salary contract by giving an NFT key to the employer after signing the contract we realize it has a bigger potential usage. Hence, a loan contract was also created. Existing Loan protocols such as AAVE work by having an ERC-20 token as collateral. The protocol allows borrowing and lending of ERC-20 tokens.

NFT are commonly used for art and collectibles. In popular websites such as Opensea, people sell and trade NFT tokens that hold information about the arts.

Creating a loan protocol for using NFT as collateral is a bit hard because each of them is unique and the value of each NFT is highly subjective. By using NFT as the key to the salary contract, we can guarantee that the NFT is at least as worthy as the funds available in the contract. This creates an opportunity for a loan protocol based on ERC721 token as collateral.

To summarize, in regards to the benefits of using smart contract to sign an employment contract, it could prevent wage theft by making it impossible for employers to break the contract intentionally. Exercising contract terms also becomes easier with smart contract. Plus, depending on the way of implementation, it will allow for a new loan protocol based on ERC-721 token.

5. Discussion

5.1 RQ1 How to implement an employment contract using a smart-contract?

Implementing an employment contract using smart contracts is a very complex task. The smart contract created in this research only touches the bare minimum condition of an employment smart contract. We are going to discuss the implication and weakness of the employment contract implemented in this research.

5.1.1 Weakness of current Salary contract

Contract only supports employment contracts with a daily salary The contract proposed in this research can only be used by a limited employment category. To deploy the salary contract, we need to provide what is the worker's daily salary. It would be really great if the contract can handle different kinds of employment contracts. But that would make the task more complex. Also, as contract logic becomes more complex, there is higher risk of vulnerabilities. In case employment salaries are implemented using smart contracts in the future, such vulnerabilities would be detrimental as all the funds for employee salary are at risk.

Contract that houses multiple salary contract Current Salary contract implemented in this research also needs to be deployed every time a new contract is issued. This may be inefficient for a really big company. To solve that, it may be better to deploy a contract that can house multiple employee contracts at once.

5.1.2 Weakness of current Loan contract

Loan contract can only handle one loan In this research, we also suggested a loan protocol that uses the Salary contract NFT key as collateral to borrow money. At this time, the logic for a loan contract can only handle one loan at a time. Also, we implemented a fixed loan repayment rate in this system. After borrowing money from the loan contract, the borrower will owe 10 percent more from the amount they had borrowed.

While this is acceptable, we can also use supply and demand rate to determine the interest rate. A more complex logic is necessary so that this kind of loan protocol can be widely adopted.

5.1.3 Employment contract upgradability

It has been known that smart contracts are immutable and cannot be changed. While the statement may be true, many methods have been used as a workaround. Such examples are by using proxy. Proxy enables a contract to be upgraded even after its deployment. It works by redirecting function calls to the latest deployed contract logic by using low-level delegate calls[8]. This method can be used to upgrade the employment contract in case something needs to be changed in the future.

5.2 RQ2 What are the benefits of using smart contract to sign employment contract?

Based on the result, we had determined that some aspects of wage theft could be tackled by using smart contract for Employment contract. But, smart contract implementation of Employment contract might solve a lot more problems regarding wage theft than that .This is because the contract can be written in a way that makes it harder for wage theft to occur. On the other hand, by making it harder for wage theft to occur, some employees might abuse the system. Some might think that the contracts are biased toward employees. Thus, the implementation needs to be implemented in a way that not just benefits the employee and employer, but acts fair against both parties.

6. Conclusion

Creating a smart contract to replace employment contract is an intricate task. While the concept of having the logic of employment contract inside smart contract does have some benefit, in the meantime it is still in the development and testing stage. Employment contract also differ greatly among different company. Some company offer monthly salary while some offer based on project accomplishment. Short term employment contract are rather simple most of the time. In contrast, permanent employee has a much more complex and longer contract due to disclaimer and what if cases. If we take insurance and tax calculation into account, the implementation will be even more complex. This further proves the point of how difficult it is to implement employment contract using smart contract is. Even if we got everything right, considering every country has different terms and conditions for employment contract. Trying to account for all the different possible outcomes and when implementing smart contract is not an easy feat. Therefore, a more extensive research and development needs to be done before smart contract will be able to replace traditional employment contract.

In this case, it may be better to have a contract employment contract standard, similar to how OpenZeppelin created a standard for ERC20 token standard for fungible token. While it may seem like an impossible task, we need to understand that even our current employment system took ages to be built and implemented. Considering that current employment contracts are susceptible to wage theft, it may be a good choice to take a step back to redesign and update our current employment and monetary system. Smart contracts are a powerful tool that might change how we sign contracts in the future.

Acknowledgment

I received a lot of help from many individuals to accomplish this research. I would like to especially thank Professor Osamu Mizuno and Assistant Professor Choi Eunjong. Without their continuous support and encouragement, I would not be able to complete this research. I am also grateful to the members of the Software Engineering Laboratory in Kyoto Institute of Technology. I am also grateful to my friends and family for their unwavering support throughout this research.

Reference

[1] N. Popper, Understanding Ethereum, Bitcoin ' s Virtual Cousin (Published 2017), (Online), URL 〈https://www.nytimes.com/2017/10/01/technology/what-is-ethereum.html〉 (Accessed on: 2009-11-18).

[2] Ethereum Virtual Machine (EVM), (Online), URL 〈https://ethereum.org/en/developers/docs/evm/〉 (Accessed on: 2009-11-18).

[3] N.F. Samreen and M.H. Alalfi, "A survey of security vulnerabilities in ethereum smart contracts," CASCON ' 20, vol.10 - 13, no.1, pp.3–7, May 2021.

[4] Non-fungible tokens (NFT), (Online), URL 〈https://ethereum.org/en/nft/〉 (Accessed on: 2022-02-09).

[5] P.J. B.A., "A statistical and geographic analysis of wage theft in hamilton county, ohio," University of Cincinnati, vol.1, no.1, pp.11–13, Aug. 2011.

[6] U.D. ofLabor, Information You Need to File a Complaint, (Online), URL 〈http://www.dol.gov/agencies/agencies/whd/contact/complaints/information〉 (Accessed on: 2022-01-15).

[7] S. Kulechov, The Aave Protocol V2, (Online), URL 〈https://medium.com/aave/the-aave-protocol-v2-f06f299cee04/〉 (Accessed on: 2022-02-07).

[8] E. Nadolinski and F. Spagnuolo, Proxy Patterns, (Online), URL 〈https://blog.openzeppelin.com/proxy-patterns/〉 (Accessed on: 2022-02-05).