Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

FL-Incentivizer: FL-NFT and FL-Tokens for Federated Learning Model Trading and Training

UMER MAJEED[1], LATIF U. KHAN[1], SHEIKH SALMAN HASSAN[1], (Student Member, IEEE) ZHU HAN[1,2], (Fellow, IEEE) AND CHOONG SEON HONG[1], (Senior Member, IEEE)

[1]Department of Computer Science & Engineering, Kyung Hee University, Yongin 17104, South Korea.

[2]Electrical and Computer Engineering Department, University of Houston, Houston, TX 77004 USA.

Corresponding author: Choong Seon Hong (e-mail: cshong@khu.ac.kr)).

ABSTRACT Federated learning (FL) is an on-device distributed learning scheme that does not require training devices to transfer their data to a centralized facility. The goal of federated learning is to learn a global model over several iterations. It is challenging to claim ownership rights and commercialize the global model efficiently and transparently. Additionally, incentives need to be provided to ensure that devices participate in the FL process. In this paper, we propose a smart contract-based framework called FL-Incentivizer, which relies on custom smart contracts to maintain flow governance of the FL process in a transparent and immutable manner. FL-Incentivizer commercializes and tokenizes the global model using FL-NFT (FL Non-Fungible Token) based on the ERC-721 standard. FL-Incentivizer uses ERC-20 compliant FL-Tokens to incentivize devices participating in FL. We present the system design and operational sequence of the FL-Incentivizer. We provide implementation and deployment details, complete smart contract codes, and qualitative evaluation of the FL-Incentivizer. After implementing FL-Incentivizer for a global iteration of a Federated learning task, we showed the FL-NFT on OpenSea and an FL-Token for a learner on MetaMask. FL-NFTs can be traded on markets such as OpenSea like other NFTs. While FL-Tokens can be transferred in the same manner as other ERC-20-based tokens.

INDEX TERMS Federated learning, Ethereum, Smart Contracts, Token, NFT, ERC-20, ERC-721, incentive, model trading, model learning, DApp

I. INTRODUCTION

Emerging wireless applications, such as haptics, digital healthcare, and intelligent transportation systems, among others generate an enormous volume of data [1], [2]. One can use such data to train machine learning (ML) models for making applications smarter. One way is to use centralized ML based on training at a centralized location. However, users as owners of isolated data are reluctant to share their private data with a centralized repository for training centralized ML models. Federated Learning (FL) [3]–[5] is collaborative on-device ML scheme that does not require training devices to transfer their data to a central facility [6]. Instead, training devices train local models on their local datasets. The local models are sent to a centralized server for collaborative training of a global model over several global iterations.

As FL is usually an iterative task, a global model (GM) on the model trading market can be more valuable if flow governance and record-keeping of the entire FL process (FLP) are open and transparent. The efficient and transparent commercialization of the intellectual property of a GM for an FL task (FLT) is a challenge. Moreover, training local models (LMs) consumes a significant amount of energy and computing resources (CPU-cycles/sec). Therefore, devices must be given an attractive incentive in response to their contribution to the global model (GM) [7].

Records keeping will be revolutionized by blockchain technology. Blockchain [8] is a distributed ledger with properties such as openness, provenance, anonymity, and transparency. Smart contracts can be executed on the blockchain to automate and govern the flow process for various applications based on corresponding transactions in an immutable and decentralized manner. Tokenization of assets can be carried out with smart contracts to verify ownership and facilitate value transfer automatically. Digital assets can be tokenized as fungible and non-fungible tokens (NFT). Intellectual property is a significant area of potential for NFT. It can be beneficial for innovators seeking to commercialize their inventions efficiently to use NFTs to promote transparency and liquidity. The most popular standard used for NFTs is ERC-721. NFTs can be categorized into dynamic NFTs and static NFTs [9]. A dynamic NFT is an NFT whose meta-data can be changed over time by external entities while the meta-data of a static NFT remains constant. The fungible tokens can serve as equivalent to monetary rewards to encourage participation from users. ERC-20 is the prevailing standard for fungible tokens.

In this paper, a dedicated smart contract is responsible to govern the FLP for an FLT. Moreover, we introduce FL Non-Fungible Token (FL-NFT) and FL fungible tokens (FL-Tokens) based on the ERC-721 standard and the ERC-20 standard, respectively. FL-NFT is a dynamic NFT and will be employed as a method to claim ownership of the GM as well as to claim royalties in the market for the usage of the GM. FL-Tokens are used to incentivise devices participating in FLP. A device can earn an FL-Token by doing a local model submission (LMS) for the FLT. Both FL-NFT and FL-Tokens are tradeable. Once a GM for an FLT is available, it can be traded as an FL-NFT on the model trading market using the ERC-721 standard. Details about the GM can be added to the FL-NFT's metadata. To the best of our knowledge, we believe that our proposal, namely FL-Incentivizer, is the first to utilize ERC-20 and ERC-721 to incentivize FL, by offering FL-Tokens and FL-NFT. Our contributions are summarized as follows:

- We propose a smart-contract-based FL-Incentivizer framework for incentivizing the devices in the FL process as well as claiming ownership rights to the GM. The FL-Incentivizer promises that all transactions, logs, and the provenance of all relevant data will be secure, immutable, and transparent.
- We introduce FL-NFTs based on the ERC-721 standard for GM trading and ownership claims.
- We introduce FL-Tokens based on the ERC-20 standard for incentivizing the partaking devices in the FL process.
- We present the comprehensive system design of the FL-Incentivizer. To manage and disseminate information about the FL process, FL-Incentivizer includes the InterPlanetary File System (IPFS). We provide the detailed operational sequence of the FL-Incentivizer which also includes the procedure to transfer FL-NFT to a new owner.
- We provide the complete implementation specifications, smart contract code [1], deployment details and qualitative evaluation for FL-Incentivizer.

The remainder of this article is laid out as follows: Section II provides a related literature review for our study. In Section III, we discuss the preliminaries relevant to our work. Section IV presents the system design and operational sequence of the FL-Incentivizer. Section V describes the implementation details, deployment setup, and qualitative evaluation of the FL-Incentivizer. In Section VI, we conclude our paper. Table. 1 lists the summary of abbreviations and notations used in this study.

[1]https://github.com/umermajeedkhu/FL-Incentivizer/tree/master/contracts

TABLE 1: Summary of Abbreviations and Notations

Abbreviation	Symbol	Description
FL	-	Federated Learning
FLT	FLT	Federated Learning Task
-	$Regulator$	Regulator of Federated learning marketplace
FLP	-	Federated Learning Process
FLTPC	$FLTPC$	Federated Learning Task Publisher Contract
FLTPCO	$FLTPCO$	FLTPC's Owner
FLTC	$FLTC$	FL-Token Contract
NFT	-	Non-Fungible Token
FL-NFT	$FLNFT$	Federated Learning Non-Fungible Token
FLNFTID	$FLNFTID$	Federated Learning Non-Fungible Token ID
FLNFTC	$FLNFTC$	Federated Learning NFT Contract
-	$FLTrainer$	The federated learning trainer (FL learning client)
IPFS	$IPFS$	InterPlanetary File System
LM	LM	Local Model
GM	GM	Global Model
GS	GS	Global Server
LMS	LMS	Local Model Submission
-	$addrof(param)$	function address_of(param) gives address of param
GI	t	Global Iteration
-	$param_i$	i^{th} entity of type param in a set
-	$param_t$	entity of type param corresponding to t^{th} GI

II. RELATED WORKS

Few works considered incentive mechanisms for participants in FL [10]–[12]. S. R. Pandey *et al.* [10] developed the two-stage Stackelberg game to reward participants based on local training accuracy levels, dropout rates, and computing costs with a Multi-Access Edge Computing (MEC) server that maximizes its utility for a fixed number of global iterations. In [11], Khan *et al.* highlighted key design aspects for incentive mechanisms to enable FL at the network edge. By using non-cooperative game theory, minimizing behavior variance among clients and minimizing model training time were the key objectives. T. Le *et al.* in [12] have devised a reverse auction-driven incentive for mobile users to participate in

FL by selling their computing resources where the base station acts as both the auctioneer and buyer. The proposed auction allows mobile users to bid based on their computation power, accuracy, and energy costs. This information is used by the base station to determine potential contributions, and then to determine the winners based on maximum social welfare. In contrast to these works focusing on the theoretical calculations of FL rewards due to clients, our study is focused on how FL rewards are awarded openly and transparently through blockchain-enabled ERC tokens. The rest of this section will discuss research using ERC-compliant tokens for a variety of application scenarios.

Crypto tokens are tradable digital assets secured through crypto wallets. [13] categorizes the crypto tokens into security tokens, payment tokens, and utility tokens. Multiple application-level standards, known as Ethereum Requests for Comment (ERC) [14], were adopted by the Ethereum community for interoperability during interaction between smart contracts as well as between Decentralized Applications (DApps) and smart contracts. ERC-20, ERC-721, ERC-777, and ERC-1155 are the most popular ERC-based token standards [13]. [15] provided an overview, protocol descriptions, available token standards, and an evaluation of NFTs. In addition, they discussed the opportunities and challenges, and potential application scenarios for the widespread adoption of NFTs. NFTs are a transparent, verifiable, tamper-resistant, and tradeable way to claim ownership of underlying physical or digital assets.

Using the provenance property in blockchain, Igarashi *et al.* in [16] used NFTs to fight deep fakes on the Internet and trace original and converted photographs. CryptoKitties [17] is the pioneer blockchain-based game. Muthe *et al.* in [18] proposes to use NFTs to tokenize in-game assets such that the objects in the game can be linked to the NFTs through metadata of NFTs. The in-game inventory of players is linked to their Ethereum wallets. Blockchain can also be utilized to tokenize virtual assets and digital characters that are independent of game service providers. Same characters or items can be used in different (but compatible) games through the crypto wallet integration. [19] used the NFTs to reward players in location-based context-aware mobile gaming such as scavenger hunts where users hunt for virtual assets in real-world locations both indoor and outdoor. These virtual assets are tradable to other players through NFT technology. Their proposal involves game developers, game players, point of interest (POI) and blockchain.

Valastin *et al.* in [20] proposed a car-sharing platform using blockchain technology. ERC-721 was used to tokenize the cars as NFT. ERC-721-based unlock tokens were issued to unlock the cars for rentees. Unlock tokens are used by an IoT device in the vehicle which verifies the unlock token on the blockchain and unlocks the car for a ride. Users are rewarded with ERC-20 tokens for using their platform. Khezr *et al.* in [21] used the NFTs for the sale of limited-edition digital artwork since this method allows the artwork auction to reach a global market, as opposed to the localized market of traditional in-person auctions. Agbesi *et al.* in [22] used the fungible tokens to represent the vote count at the polling station in the blockchain based e-voting framework. The proposed scheme allows polling stations to produce a fungible token that represents the vote count at the polling station. This is so that when the tokens are transmitted to the next collation point, they cannot be tempered. Thus, preventing counting fraud at collation points.

M. Stefanovic *et al.* in [23] uses ERC-721 and ERC-20 to optimize transfers in land administration systems. Land administration involves managing ownership history as well as physical, spatial and topographic information about real estate. ERC-721 based NFT represents a complete property. While, with the help of ERC-20 tokens, multiple entities can own a single property, and then can transfer 100% or fewer shares to the new owner. R.W. Ahmad *et al.* in [24] analyzed the use of blockchain in smart city waste management systems. They discussed Recerreum which is an Ethereum-enabled project to reward citizens by incentivizing them with ERC-20 based tokens for sorting and depositing domestic waste in appropriate vending machines at waste collecting centers.

III. PRELIMINARIES

The following section presents an overview of the technologies and platforms used to design the FL-Incentivizer framework and to implement and evaluate it.

A. DECENTRALIZED APPLICATIONS

Software application is a piece of software designed to accomplish a specific purpose. Usually, centralized applications are managed by a single entity. Unlike centralized applications, Decentralized Applications (DApps) do not have a single administrator or controller. Modern DApps are self-governing applications and are composed of a front-end interface, an application programming interface (API), and a back-end. Smart contracts serve as the back-end of DApps to execute the business logic with integrated access control and store the state on the blockchain [25]. The front-end UI of DApps may be rendered via a centralized server or a decentralized file storage network like IPFS [26], but the back-end of a decentralized application is always decentralized.

B. ETHEREUM AND SMART CONTRACTS

Ethereum [27] is a decentralized platform that runs byte-code compiled by scripting languages on a Turing-complete virtual machine. Smart contracts [28], [29] were devised in 1994 by Nick Szabo [30], [31]. According to its original conception, smart contracts were intended to embed contractual stipulations in hardware and software, making violating the agreement costly to the breaching party [32]. Incorporating smart contracts into the blockchain offers the possibility of implementing a wide range of applications [33]. Smart contracts are typically implemented using Solidity, a high-level programming language [34]. A smart contract is executed on Ethereum whenever it is triggered by transactions via DApps.

There is a unique private key associated with each Externally Owned Account (EOA) for signing transactions [35].

C. METAMASK

MetaMask [36] is an internet-browser-extension-based Ethereum wallet. MetaMask enables users to hold EOAs and sign transactions. Wallet generation algorithms enable MetaMask to generate multiple EOAs off-chain. Each EOA has a private key and a public address. MetaMask also provides a seed phrase to recover EOAs in case of loss of password. The seed phrase and private keys should be kept secret. Most DApps connect to MetaMask automatically, following the user's approval. It allows users to see their assets, such as ETH and Tokens, and transaction history. Users can import and export private keys for EOAs.

D. HARDHAT

Hardhat [37] is an "Ethereum development environment for professionals" to compile, deploy, debug, and test Solidity smart contracts on Ethereum-based local networks, public testnets, and mainnets. With Hardhat, developers can also automate the repetitive tasks entailed in the development of smart contracts and DApps [38]. A local Ethereum network built specifically for developers is part of Hardhat. Hardhat functionality is centered on Solidity debugging tools, such as stack traces, console.log(), and error messages for failed transactions.

E. ERC-20 AND FUNGIBLE TOKENS

ERC-20 is the first standard for fungible tokens to define a standardized interface. T. The term "fungible token" refers to the fact that they are interchangeable and mutually indistinguishable. Therefore, they can be exchanged for tokens of the same type and value. Some popular ERC-20 tokens are Tether (USDT), Shiba Inu (SHIB), Chainlink (LINK), and MATIC. To govern token creation, exchange, and valuation, ERC-20 defines two standard events besides nine standard interfaces [39], [40]. The function names, parameter types, and return values must strictly adhere to ERC-20 standards. The minimum set of required functions by ERC-20 are transfer, transferFrom, totalSupply, allowance, approve, and balanceOf.

F. ERC721 AND NON-FUNGIBLE TOKENS

The ERC-721 token standard is applicable to NFT [41]. A NFT is indivisible, unique, and not interchangeable with any other token [42]. Consequently, NFTs can tokenize physical [43] or digital assets and identify them uniquely. In essence, a NFT is a record that includes several fields. These are the address of the NFT's 'wallet owner', the NFT's ID, the Metadata Hash or possibly a link to a repository that contains the Metadata. By relying on the cryptography of the underlying blockchain, an NFT provides openness, legitimacy, transparency, reliability, ownership, uniqueness (scarcity), globalization, and constancy for an asset. In order for NFTs to be interoperable with any NFT-based platform, a standard interface ERC-721 is specified. A method called tokenURI is included in ERC-721 to locate the metadata for a given NFT.

G. TESTNETS

Smart contracts in production are deployed on public networks called mainnets. However, deployment on mainnets is costly. For cost-effective validation, verification, and testing of smart contracts [38]: local networks, and public testnets can be used. Similar to mainnets, testnets have their own collection of nodes running the same protocol as the respective mainnet. The popular Ethereum public testnets are Rinkeby [44] and Ropsten [45]. The native currency, tokens, and NFTs on testnets have no market value. Testnets generally have a lower transaction fee than mainnets. Each testnet has several faucets, which can transfer native currency, tokens, and NFTs to users as a freebie.

H. FEDERATED LEARNING

FL is a collaborative technique for training a shared model from geographically distributed local datasets with additional computation at the edge [46], [47]. The FL server consolidates LM updates from learners for each global iteration (GI) to release the updated GM [11]. Below is a formulation of the FLP:

Consider that an artificial neural network (ANN) can classify input data into C classes, then we define $[C] = \{1,...,C\}$. Let $\dagger(\boldsymbol{\omega})$ be the output function of the ANN and $\boldsymbol{\omega}$ are the weights of the ANN. $\mathcal{X}$ is a compact Euclidean feature space for the input data. $[X]$ can be mapped to the $[Y] = [C]$ label space by $\dagger(\boldsymbol{\omega})$. If the probability that class q is mapped for data-point $x \in \mathcal{X}$ is given by $p_q(\boldsymbol{x}, \boldsymbol{\omega})$. Then for a data-point $\{x, y\}$ with one-hot encoded label, the cross-entropy loss is [47], [48]:

$$f_r(\boldsymbol{\omega}) = -\sum_{q=1}^{C} \mathbb{1}_{y=q} \log p_q(\boldsymbol{x}, \boldsymbol{\omega}). \tag{1}$$

Let $\mathcal{D}_i$ be the data-set of the i^{th} client $(FLTrainer_{i,t+1})$ in GI $t+1$. The total data points in $\mathcal{D}_i$ are $n_i = |\mathcal{D}_i|$. F_k denotes the local loss and is formulated as [47]:

$$F_i(\boldsymbol{\omega}) = \frac{1}{n_i} \sum_{r \in D_i} f_r(\boldsymbol{\omega}). \tag{2}$$

The local gradient is then calculated as follows:

$$g_i = \nabla F_i(\boldsymbol{w}_t) \quad \text{where} \quad \delta_i = |D_i|\, g_i. \tag{3}$$

LM weights are updated at GI $t+1$ as follows:

$$\boldsymbol{w}_{t+1}^{i} \leftarrow \boldsymbol{w}_t - \eta g_i, \quad \forall i. \tag{4}$$

Let ψ_{t+1} be the set of FL clients for GI $t+1$. Then,

$$\psi_{t+1} = \bigcup FLTrainer_{i,t+1}, \forall i \tag{5}$$

In FL settings, we calculate the global loss as follows:

$$f(\boldsymbol{w}) = \sum_{i \in \psi_{t+1}} \frac{n_i}{n} F_i(\boldsymbol{w}). \tag{6}$$

Subsequently, global gradient is calculated as [47]:

$$\nabla F\left(\boldsymbol{w}_t\right) = \sum_{i\in\psi_{t+1}} \frac{n_i}{n} g_i = \frac{\sum_{i\in\psi} \delta_i}{\sum_{i\in\psi_{t+1}} |\mathcal{D}_i|}. \qquad (7)$$

The GM weights are updated at GI $t+1$ by utilizing Federated averaging (FedAvg) [46], [47] as:

$$\boldsymbol{w}_{t+1} \leftarrow \sum_{i\in\psi_{t+1}} \frac{n_i}{n} \boldsymbol{w}^i_{t+1} \qquad (8)$$

or

$$\boldsymbol{w}_{t+1} \leftarrow \boldsymbol{w}_t - \eta \nabla F\left(\boldsymbol{w}_t\right). \qquad (9)$$

During this FLP, the overall objective is to minimize the global loss as follows [49]:

$$\min_{\boldsymbol{w}} f(\boldsymbol{w}). \qquad (10)$$

The FL-Incentivizer framework utilizes the aforementioned technologies to optimize its system design, sequence operation, and implementation. In particular, FL is used to update the GM which is tokenized as NFT via ERC-721, learners get the reward through ERC-20 based tokens. The smart contracts are deployed and tested on Ethereum testnets via Hardhat. The entities in FL-Incentivizer also interact with smart contracts by signing transactions in DApp via the MetaMask wallets.

IV. PROPOSED FRAMEWORK

In this section, we describe the system design and operational sequence of the FL-Incentivizer framework. For flow governance in federated learning settings, FL-Incentivizer uses customized smart contracts. To commercialize the global model and to incentivize learners in a federated learning process, FL-Incentivizer relies on ERC-721-based tokens and ERC-20-based tokens respectively.

The system design of the FL-Incentivizer framework is shown in Fig. 1. The system design consists of three layers namely, the managerial layer, the distributed decentralized layer, and the client layer. This system supports a *Federated Learning Task* (FLT), where FLT is a deep learning or ML task to train a GM using FL algorithms such as FedAvg or FedProx. We denote the new FLT as FLT. For simplicity, onwards we will consider only this particular FLT unless mentioned otherwise as the process for all FLTs will be the same. GM_t denotes the GM for the t^{th} GI of FLT.

The managerial layer includes a regulator, the FLTPCO, and a Global Server. The regulator is the body responsible for standardizing the FL marketplace and ecosystem. The regulator is in charge of the deployment of FLNFTC. The regulator is also responsible for standardizing the meta-data for NFTs in FLNFTC. The regulator is denoted as $Regulator$ in our study. The FL Task Publisher Contract's owner (FLTPCO) is a person/company who owns an FLTPC. When a person/company wants to train an FL model, it will deploy a new dedicated FLTPC. When an FL-NFT is traded, the new owner of the FL-NFT will also own the corresponding FLTPC. We denote the FLTPCO corresponding to FLT as $FLTPCO$. The Global Server (GS) is run by $FLTPCO$. It is responsible for aggregating the LMs to the GM for each iteration for the FLT. It is denoted as GS in our study.

The distributed decentralized layer consists of FLNFTC, FLTPC, FLTC, and IPFS. The FL-NFT Contract (FLNFTC) is deployed for the whole FL market by the $Regulator$. The whole framework has only one FLNFTC. It allows anyone to mint a new FL-NFT for any FL task. The FLNFTC is based on the ERC-721 standard and is denoted as $FLNFTC$. The FL Task Publisher Contract (FLTPC) is deployed by a person/company who wants to train an FL model. A new FLTPC is deployed for every new FL task. The FLTPC is responsible for orchestrating the whole FLP and is denoted as $FLTPC$. The tokenId of $FLNFT$ in $FLTPC$ is denoted as $FLTPC.FLNFTID$. The FL-Token Contract (FLTC) is the contract deployed by the corresponding FLTPC at the start of the FLP. The FLTC is based on the ERC-20 standard and is denoted as $FLTC$. Each FLT has its seperate FLTC. The corresponding FLTPC is responsible for minting new FL-Tokens in the FLTC for the learners. As the FLP progresses, new FL-Tokens are awarded to participants. IPFS is a decentralized file storage system that provides users with content identifiers (CIDs) for content stored on it. An IPFS system is a network of content-addressable storage which means if two entities upload the same content, the CIDs will be the same. We denote the IPFS as $IPFS$.

The client layer consists of several FL-Trainers. An FL-Trainer is the partaking device/client in the FLP for FLT. $FLTrainer_{i,t+1}$ denotes the i^{th} client in the $t+1^{th}$ GI of FLT. FL-Trainer downloads the $(t)^{th}$ GM of FLT and generates the LM by using its local dataset $D_{i,t+1}$.

In addition to these entities, the system design involves FL-NFT and FL-Tokens. FL-NFT is a dynamic NFT based on the ERC-721 standard. Each FL-NFT is related to a particular FLT. We denote the FL-NFT related to FLT as $FLNFT$. Each FL-NFT is unique and assigned the Uniform Resource Identifier (URI) denoted as $tokenURI$ and pointing to the meta-data of the latest GM of the FLT. The FL-NFT is also assigned the IPFS hash (denoted as $GMipfsHash$) of the latest GM of the FLT. The address of the corresponding $FLTPC$ is also stored and denoted as $FLTPCAddress$. The $tokenURI$, $GMipfsHash$, and $FLTPCAddress$ for all FL-NFTs are unique. The FL-NFTs themselves can be traded and are an incentive for task publishers. Generally, FL-NFT is owned by $FLTPCO$ through its crypto wallet. An FL-Token is an ERC-20-based token awarded to devices participating in FLP. All FL-Tokens minted by the same FLTC are fungible to each other. We denote an FL-Token as $FLToken$. An FL-Token is generally owned by FL-Trainer through its crypto wallet.

The aforementioned entities are an integral part of FL-Incentivizer. Now, our discussion will focus on the operational sequence of the FL-Incentivizer as illustrated in Fig. 2. The simplified sequence steps are as follows:

- Step 1: The $Regulator$ deploys the $FLNFTC$ for the FL market. The owner of the $FLNFTC$ is set to

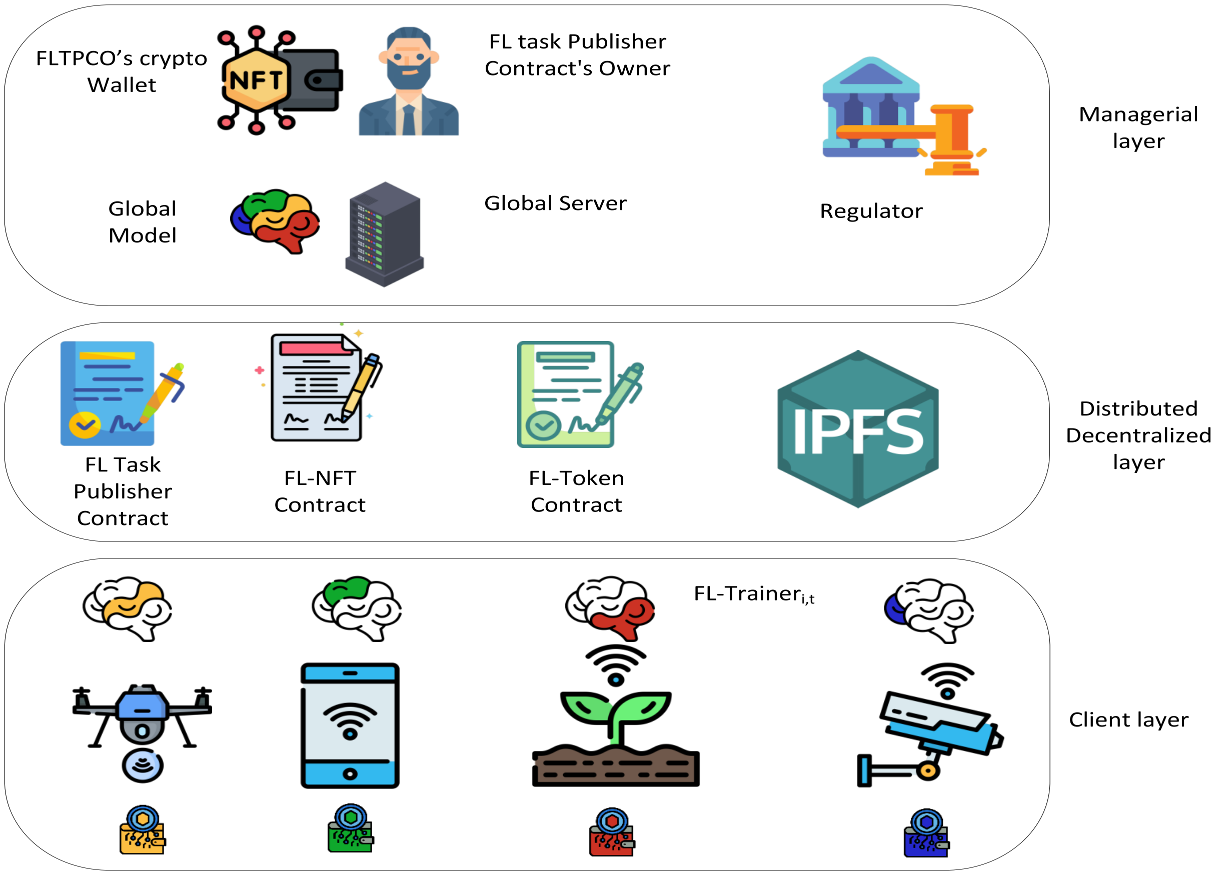


FIGURE 1: FL-Incentivizer: System Design

the address of $Regulator$. This step is summarized in procedure $DEPLOY\_FLNFTC$ of Algorithm 1.

- Step 2: For the FLT, $FLTPCO$ will deploy the $FLTPC$. The owner of the $FLTPC$ is set to the address of $FLTPCO$. Moreover, $FLTPC$ is also paused. The paused status of $FLTPC$ means that most of the $FLTPC$ operations are not yet ready. This step is summarized in procedure $DEPLOY\_FLTPC$ of Algorithm 2.
- Step 3: During the deployment of FL task Publisher Contract $FLTPC$. The $FLTPC$ will deploy the FL-token Contract $FLTC$ for FLT. The owner of $FLTC$ is set to the address of $FLTPC$. This step is summarized in procedure $DEPLOY\_FLTC$ of Algorithm 4.
- Step 4: The $FLTPCO$ starts the procedure $CREATE\_FLNFT$ in Algorithm 2. In this procedure, the $FLTPCO$ generates the initial GM weights for FLT and uploads it to IPFS. The resulting hash is $GMipfsHash$. This becomes GM_t where $t = 0$. The $FLTPCO$ creates the related files for FLT such as FL task guidelines, LMS guidelines, LMS reward criteria, and any other custom details; and uploads them to IPFS. The $FLTPCO$ also creates a representing image for FL-NFT, which is uploaded to IPFS. These details are added to a JSON-formatted meta-data file. The meta-data also includes the addresses of $FLTPC$, $FLTC$, and $FLNFTC$. The meta-data is denoted as $GM\_Meta\_Data_t$. A sample meta-data file can be seen from Appendix A. The $GM\_Meta\_Data_t$ is uploaded to IPFS. The resulting hash is $tokenURI$. Afterwards, the $FLTPCO$ initiates procedure $createFLNFT$ in Algorithm 4. Subsequently, the $FLTPC$ mints the $FLNFT$ on $FLNFTC$ for $FLTPCO$ using procedure $mintFLNFT$ in Algorithm 6.
- Step 5: The $FLTPCO$ then starts the procedure $START\_LMSubmission$ in Algorithm 2 to start the LMSs on the $FLTPC$. This initiates procedure $start\_LMSs$ in Algorithm 4. This procedure checks if $FLTPC.LMSAccepting$ is set to false or not. The $FLTPC.LMSAccepting == true$ indicates that $FLTPC$ is currently accepting LMSs and $FLTPC.LMSAccepting == false$

Algorithm 1 : Regulator $Regulator$

```
1: procedure DEPLOY_FLNFTC
2:     Deploy $FLNFTC$
3:         Set $FLNFTC.owner = addrof(Regulator)$
4: end procedure
```

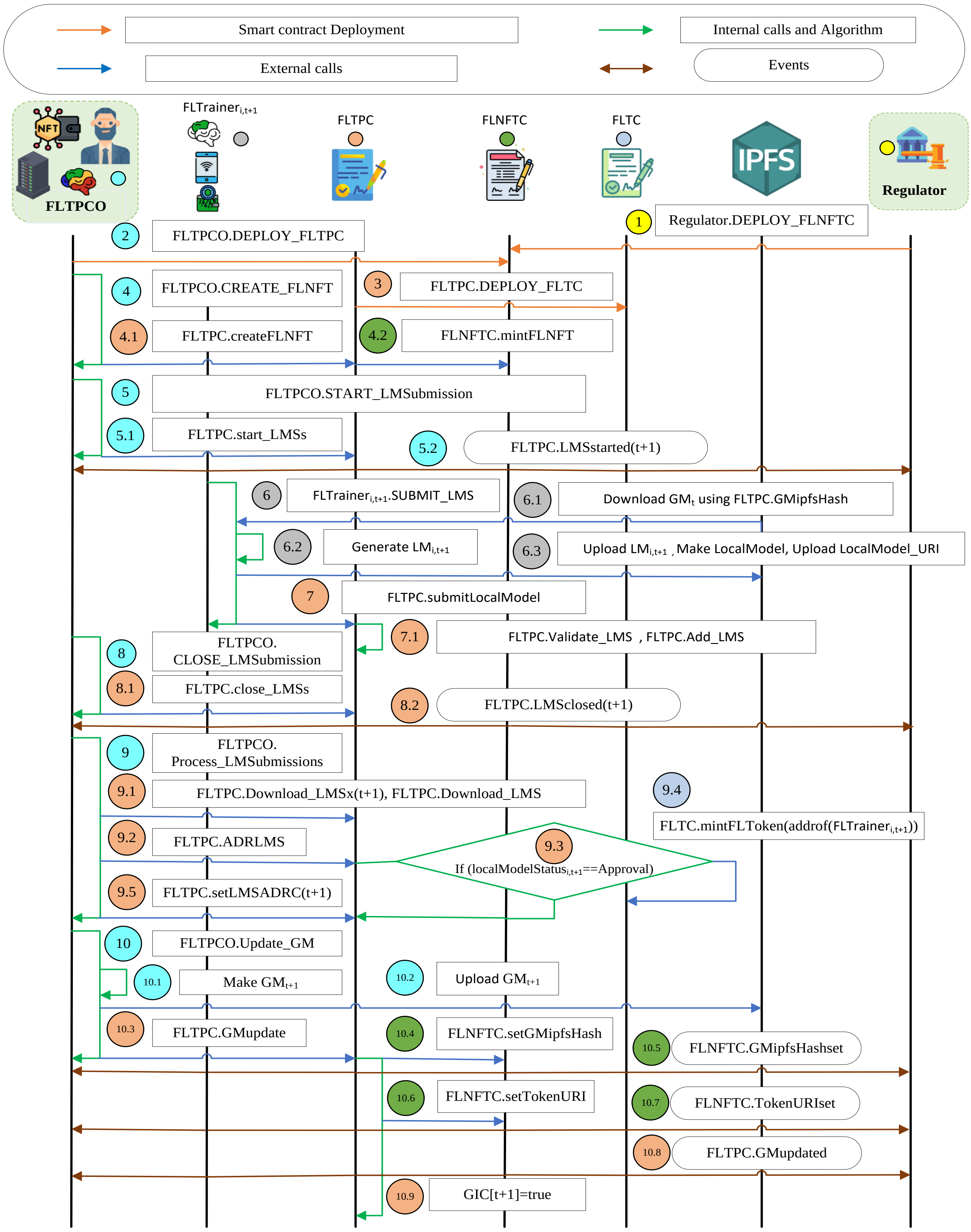

Smart contract Deployment
Internal calls and Algorithm
External calls
Events
FLTPCO
FLTrainer$_{i,t+1}$
FLTPC
FLNFTC
FLTC
IPFS
Regulator
1 Regulator.DEPLOY_FLNFTC
2 FLTPCO.DEPLOY_FLTPC
3 FLTPC.DEPLOY_FLTC
4 FLTPCO.CREATE_FLNFT
4.1 FLTPC.createFLNFT
4.2 FLNFTC.mintFLNFT
5 FLTPCO.START_LMSubmission
5.1 FLTPC.start_LMSs
5.2 FLTPC.LMSstarted(t+1)
6 FLTrainer$_{i,t+1}$.SUBMIT_LMS
6.1 Download GM$_t$ using FLTPC.GMipfsHash
6.2 Generate LM$_{i,t+1}$
6.3 Upload LM$_{i,t+1}$, Make LocalModel, Upload LocalModel_URI
7 FLTPC.submitLocalModel
7.1 FLTPC.Validate_LMS , FLTPC.Add_LMS
8 FLTPCO. CLOSE_LMSubmission
8.1 FLTPC.close_LMSs
8.2 FLTPC.LMSclosed(t+1)
9 FLTPCO. Process_LMSubmissions
9.1 FLTPC.Download_LMSx(t+1), FLTPC.Download_LMS
9.2 FLTPC.ADRLMS
9.3 If (localModelStatus$_{i,t+1}$==Approval)
9.4 FLTC.mintFLToken(addrof(FLTrainer$_{i,t+1}$))
9.5 FLTPC.setLMSADRC(t+1)
10 FLTPCO.Update_GM
10.1 Make GM$_{t+1}$
10.2 Upload GM$_{t+1}$
10.3 FLTPC.GMupdate
10.4 FLNFTC.setGMipfsHash
10.5 FLNFTC.GMipfsHashset
10.6 FLNFTC.setTokenURI
10.7 FLNFTC.TokenURIset
10.8 FLTPC.GMupdated
10.9 GIC[t+1]=true

FIGURE 2: FL-Incentivizer: Operational Sequence

Algorithm 2 : FL Task Publisher Contract's Owner $FLTPCO$

```
1: procedure $DEPLOY\_FLTPC$
2:     Deploy $FLTPC$
3:         Set $FLTPC.owner = addrof(FLTPCO)$
4:         Call $FLTPC.pause()$
5:         Call $FLTPC.DEPLOY\_FLTC$
6: end procedure

   Input: $FLTPC$
1: procedure $CREATE\_FLNFT$
2:     Make $GM_t$                                  ▷ $t = 0$
3:     $GMipfsHash$ = Upload $GM_t \to IPFS$
4:     Make $GM\_Meta\_Data_t$ for $GM_t$
5:     $tokenURI$ = Upload $GM\_Meta\_Data_t \to IPFS$
6:     if $(FLTPC.FLNFTID == 0)$ then
7:         Call $FLTPC.createFLNFT(GMipfsHash, tokenURI)$
8:     end if
9: end procedure

   Input: $FLTPC$
1: procedure $START\_LMSubmissions$
2:     Call $FLTPC.start\_LMSs$
3: end procedure

   Input: $FLTPC$
1: procedure $Close\_LMSubmissions$
2:     Call $FLTPC.close\_LMSs$
3: end procedure

   Input: $FLTPC, t+1$
1: procedure $Process\_LMSubmissions$
2:     foreach $FLTrainer_{i,t+1}$ in FLTPC.Download_LMSx($t+1$)
3:         $FLTrainer_{i,t+1}$ = Call FLTPC.Download_LMS($t+1$, $addrof(FLTrainer_{i,t+1})$)
4:         Call $FLTPC.ADRLMS(addrof(FLTrainer_{i,t+1}), t+1, LMStatus_{i,t+1})$
5:     end foreach
6:     Call $FLTPC.setLMSADRC(t+1)$
7: end procedure
```

indicates that LMSs are closed in $FLTPC$. If $FLTPC.LMSAccepting$ is false, the respective procedure changes $FLTPC.LMSAccepting$ to true and emits the event $LMSstarted(t+1)$ where $t+1$ is GI for which LMSs are just started. FL-Trainers listen for event $LMSstarted(t+1)$ for submitting their LM updates.

- Step 6: The $FLTrainer_{i,t+1}$ then starts the procedure $SUBMIT\_LMS$ in Algorithm 7 to start the LMSs on $FLTPC$. For this, the $FLTrainer_{i,t+1}$ gets the latest GM hash $FLTPC.GMipfsHash$

Algorithm 3 : FL Task Publisher Contract's Owner $FLTPCO$ - Continued

```
   Input: $FLTPC, t+1$
1: procedure $Update\_GM$
2:     Make $GM_{t+1}$ using (8)
3:     $GMipfsHash$ = Upload $GM_{t+1} \to IPFS$
4:     Make $GM\_Meta\_Data_{t+1}$ for $GM_{t+1}$
5:     $tokenURI$ = Upload $GM\_Meta\_Data_{t+1} \to IPFS$
6:     Call $FLTPC.GMupdate(GMipfsHash, tokenURI)$
7: end procedure
```

from $FLTPC$, downloads the latest GM GM_t from $IPFS$ using $FLTPC.GMipfsHash$, and calculates the local model $LM_{i,t+1}$ on local dataset $D_{i,t+1}$ using [4]. The $FLTrainer_{i,t+1}$ then uploads $LM_{i,t+1}$ to $IPFS$ to get $LMipfsHash$. The $FLTrainer_{i,t+1}$ also makes Local_Model_URI which is meta-data JSON for $LM_{i,t+1}$. The $FLTrainer_{i,t+1}$ then uploads Local_Model_URI to $IPFS$ to get $LModelURI$. Afterwards, $FLTrainer_{i,t+1}$ calls $FLTPC.submitLocalModel$ to submit its LM to $FLTPC$. The $FLTPC$ may have hard-coded the limit for the number of LMSs for GI $t+1$ allowed.

- Step 7: The procedure $submitLocalModel$ in Algorithm 4 is triggered by $FLTrainer_{i,t+1}$. The local model submission $LMS_{i,t+1}$ is validated by FLTPC.Validate_LMS function. The FLTPC.Validate_LMS function may not accept the LMS if the limit has been reached for the $t+1$ GI. If the LMS is valid, the $LMS_{i,t+1}$ is added to the LMSs against the GI $t+1$ and the address of $FLTrainer_{i,t+1}$ denoted as $addrof(FLTrainer_{i,t+1})$ by function FLTPC.Add_LMS. The LM status $LMS_{i,t+1}.LMStatus$ is set as Submitted.
- Step 8: The $FLTPCO$ closes the LMSs for current GI by procedure $CLOSE\_LMSubmission$ of Algorithm 2. This initiates procedure $close\_LMSs$ in Algorithm 4. This procedure checks if $FLTPC.LMSAccepting$ is set to false or not. If $FLTPC.LMSAccepting$ is true, this procedure changes $FLTPC.LMSAccepting$ to false, sets $LMSC[t+1]$ to true to indicate LMSs are closed for $t+1$, and emits the event $LMSclosed(t+1)$ where $t+1$ is GI for which LMSs are just closed. The FL-Trainers listen to the event $LMSclosed(t+1)$ and drop further LMSs.
- Step 9: The $FLTPCO$ start the procedure $Process\_LMSubmissions$ in Algorithm 2. In this procedure, $FLTPCO$ first downloads the addresses of LM submitters by invoking function Download_LMSx($t+1$). Afterwards, for each $FLTrainer_{i,t+1}$ in FLTPC.Download_LMSx($t+$

Algorithm 4 : FL Task Publisher Contract $FLTPC$

```
1: procedure DEPLOY_FLTC
2:     Deploy $FLTC$
3:         Set $FLTC.owner = addrof(FLTPC)$
4: end procedure

   Input: $tokenURI, GMipfsHash$
1: procedure createFLNFT
2:     $tokenId$ = call $FLNFTC.mintFLNFT$
3:     Set $FLTPC.FLNFTID$ = $tokenId$
4: end procedure

1: procedure start_LMSs
2:     if $FLTPC.LMSAccepting == false$ then
3:         Set $FLTPC.LMSAccepting = true$
4:         Emit $FLTPC.LMSstarted(t+1)$;
5:     end if
6: end procedure

   Input: $LMipfsHash$, $LModelURI$, $t+1$, $addrof(FLTrainer_{i,t+1})$
1: procedure submitLocalModel
2:     if FLTPC.Validate_LMS($LMipfsHash$, $LModelURI$, $t+1$, $addrof(FLTrainer_{i,t+1})$) then
3:         Call FLTPC.Add_LMS($LMipfsHash$, $LModelURI, t+1, addrof(FLTrainer_{i,t+1}$ )
4:     end if
5: end procedure

1: procedure close_LMSs
2:     if $FLTPC.LMSAccepting == true$ then
3:         Set $FLTPC.LMSAccepting = false$
4:         Emit $LMSclosed(t+1)$
5:         Set $LMSC[t+1] = true$;
6:     end if
7: end procedure

   Input: $addrof(FLTrainer_{i,t+1})$, $t+1$, $LMStatus_{i,t+1}$
1: procedure ADRLMS
2:     if $LMStatus_{i,t+1} == Approval$ then
3:         Call $FLTC.mintFLToken($ $addrof(FLTrainer_{i,t+1}))$
4:         Set $LMS_{i,t+1}.LMStatus == Rewarded$
5:     else
6:         Set $LMS_{i,t+1}.LMStatus == Denied$
7:     end if
8: end procedure
```

1), $FLTPCO$ downloads the corresponding the LMS denoted as $LMS_{i,t+1}$ by calling FLTPC.Download_LMS($t+1$,$FLTrainer_{i,t+1}$). The $FLTPCO$ checks $LMS_{i,t+1}$ and approves or denies it by calling procedure $FLTPC.ADRLMS$ of Algorithm 4. Where $LMStatus_{i,t+1} \in \{Approved, Denied\}$ is the respective decision for $LMS_{i,t+1}$ and ADRLMS stands for Approval, Disapproval, Reward for LMS. If the $LMS_{i,t+1}$ is "Approved", $FLTPC.ADRLMS$ invokes procedure $FLTC.mintFLToken$ of Alogorithm 8 to mint FL-Token for the corresponding $FLTrainer_{i,t+1}$. Lastly, $FLTPCO$ executes procedure $FLTPC.setLMSADRC(t+1)$ of Algorithm 5 which sets the $LMSADRC(t+1)$ flag for GI $t+1$ to indicate that the LMSs for respective GI have been approved, denied and rewarded.

Algorithm 5 : FL Task Publisher Contract $FLTPC$ - Continued

```
   Input: $t+1$
1: procedure setLMSADRC
2:     Set $LMSADRC[t+1] = true$
3: end procedure

   Input: $t+1, GMipfsHash, tokenURI$
4: procedure GMupdate
5:     setGMipfsHashF = Call $FLNFTC.setGMipfsHash($ $GMipfsHash, FLNFTID)$
6:     setTokenURIF = Call $FLNFTC.setTokenURI($ $tokenURI, FLNFTID)$
7:     if setTokenURIF && setGMipfsHashF then
8:         Emit $GMupdated(t+1, GMipfsHash,$ $tokenURI)$;
9:         Set $FLTPC.tokenURI$ = $tokenURI$
10:        Set $FLTPC.GMipfsHash$ = $GMipfsHash$
11:        Set $GIC[t+1] = true$
12:    end if
13: end procedure
```

- Step 10: The $FLTPCO$ starts the procedure $Update\_GM$ in Algorithm 3. Let's denote the approved LMSs in the previous steps as $LMS_{\hat{i},t+1}$. The $FLTPCO$ computes the GM_{t+1} using [8], and uploads it to IPFS. The resulting hash is $GMipfsHash$. The $FLTPCO$ also updates the representing image for FL-NFT, which is uploaded to IPFS. These details are updated in the meta-data JSON file. The meta-data of GM_{t+1} for GI $t+1$ is denoted as $GM\_Meta\_Data_{t+1}$. The $GM\_Meta\_Data_{t+1}$ is uploaded to IPFS. The resulting hash is $tokenURI$. Subsequently, the $FLTPCO$ triggers procedure $FLTPC.GMupdate$ in Algorithm 5. The $FLTPC.GMupdate$ calls procedure $FLNFTC.setGMipfsHash$ in Algorithm 6 and procedure $FLTPC.setTokenURI$ in Algorithm 6 to set $GMipfsHash$ and $tokenURI$ of $FLNFT$ respectively. It is necessary to mention that only registered $FLTPC$ (at the minting of

Algorithm 6 : FL-NFT Contract $FLNFTC$

```
Input: $tokenURI$, $addrof(FLTPC)$, $GMipfsHash$, $addrof(FLTPCO)$
Output: $tokenId$
1: procedure mintFLNFT
2:     $tokenId$ = Mint FL-NFT for $FLTPCO$
3:     Set $FLNFT.tokenURI$ = $tokenURI$
4:     Set $FLNFT.GMipfsHash$ = $GMipfsHash$
5:     Set $FLNFT.FLTPC$ = $addrof(FLTPC)$
6: end procedure

Input: $GMipfsHash$, $FLNFTID$
1: procedure setGMipfsHash
2:     if FLTPC.Validate_GMipfsHash($GMipfsHash$, $FLNFTID$ then
3:         Set $GMipfsHashes[FLNFTID]$ = $GMipfsHash$
4:         Ascertain Unique $GMipfsHashes$
5:         Emit $GMipfsHashset(FLNFTID, GMipfsHash)$
6:         Return true
7:     else
8:         Revert
9:     end if
10: end procedure

Input: $tokenURI$, $FLNFTID$
1: procedure setTokenURI
2:     if FLTPC.Validate_TokenURI($tokenURI$, $FLNFTID$ then
3:         Set $tokenURIs[FLNFTID] = tokenURI$;
4:         Ascertain Unique $tokenURIs$
5:         Emit $TokenURIset(FLNFTID, tokenURI)$
6:         Return true
7:     else
8:         Revert
9:     end if
10: end procedure
```

Algorithm 7 : FL-Trainer $FLTrainer_{i,t+1}$

```
Input: $FLTPC$, $t$
1: procedure SUBMIT_LMS
2:     Get $FLTPC.GMipfsHash$
3:     Download $GM_t \leftarrow IPFS$ using $FLTPC.GMipfsHash$
4:     Generate $LM_{i,t+1}$ using (4)
5:     $LMipfsHash$ = Upload $LM_{i,t+1} \rightarrow IPFS$
6:     Make Local_Model_URI for $LM_{i,t+1}$
7:     $LModelURI$ = Upload Local_Model_URI $\rightarrow IPFS$
8:     Call $FLTPC.submitLocalModel(LMipfsHash, LModelURI, t+1)$
9: end procedure
```

Algorithm 8 : FL Token Contract $FLTC$

```
Input: $addrof(FLTrainer_{i,t+1})$
1: procedure mintFLToken
2:     d = Call $FLTC.decimals()$
3:     Mint $1 * 10^{18}$ FLT for $addrof(FLTrainer_{i,t+1})$
4: end procedure
```

Algorithm 9 : FL-NFT's transfer

```
Executor: $current\_FLTPCO$
Input: $new\_FLTPCO$
1: procedure FLTPCO_Transfer
2:     Call $FLTPC.close\_LMSs$
3:     Pause $FLTPC$
4:     Transfer $FLNFT$ to $new\_FLTPCO$
5: end procedure

Executor: $new\_FLTPCO$
1: procedure FLTPCO_Unpause
2:     Unpause $FLTPC$
3: end procedure
```

$FLNFT$) can execute $FLNFTC.setGMipfsHash$ and $FLNFTC.setTokenURI$ for particular $FLNFTID$. The $FLNFTC.setGMipfsHash$ then calls function FLNFTC.Validate_GMipfsHash to validate the submitted $GMipfsHash$ and subsequently set the $GMipfsHash$ of $FLNFT$. The function FLNFTC.Validate_GMipfsHash executes a code to ascertain the uniqueness of $GMipfsHash$ for all FL-NFTs. Similarly, the $FLNFTC.setTokenURI$ invokes function FLNFTC.Validate_TokenURI to validate the submitted $tokenURI$ and subsequently set the $tokenURI$ of $FLNFT$. The function FLNFTC.Validate_TokenURI execute a code to ascertain the uniqueness of $tokenURI$ for all FL-NFTs. It is necessary to mention that all FL-NFTs must have unique GMipfsHashes and tokenURIs at a time. Afterwards, the $FLTPC$ emits the event $FLTPC.GMupdated$. Finally, the $FLTPC$ flags the $GIC[t+1]$ to indicate that the GI $t+1$ is completed.

Step 1 is done only once for the whole FL model trading market. However, for every instance of FLT, Steps 2-4 of the above sequence are repeated to set up the FLP environment. While, for each GI $t+1$ of FLT, Steps 5-10 are repeated. FL-NFT trading involves transferring the FL-NFT from its current owner to the buyer. To transfer the $FLNFT$ to a new owner, the $FLTPCO\_Transfer$ procedure in Algorithm 9 is started by the current owner, which first closes the LMSs by calling $FLTPC.close\_LMSs$. Then, it pauses the $FLTPC$. Afterward, it transfers the $FLNFT$ to the new owner. The new owner may start the procedure $FLTPCO\_Unpause$ in Algorithm 9 to unpause $FLTPC$.

TABLE 2: Parameters

Parameter	**Description**	**Rinkeby**	**Ropsten**
$addrof(Regulator)$	address of $Regulator$	0x8fa37ECF3d89361e60E7e6adf55485a e62cd72b2	0x8fa37ECF3d89361e60E7e6adf55485a E62cD72B2
$addrof(FLTPCO)$	address of $FLTPCO$	0xa0969AeA747c336b49256CFC4Cc2F 6E265F6B722	0xa0969AeA747c336b49256CFC4Cc2F 6E265F6B722
$addrof(FLTrainer_1)$	address of $FLTrainer_1$	0xA1Ee512DBC5b437eeb317C1fBc2E6 19648cA5d33	0xA1Ee512DBC5b437eeb317C1fBc2E6 19648cA5d33
$addrof(FLTPC)$	address of $FLTPC$	0xdC63FB6f6a11c358D06Cf05A1049e C04cf1e5bc1	0x1d9Cebd90Aa66068cD9FD3d75479D bDeDA65ebeB
$addrof(FLNFTC)$	address of $FLNFTC$	0x620Ef4E1cDE7f1841538279E6baf1d 1b96f6e6c4	0xd230D92fdd47e3732CeD57eD76501d 2822172665
$addrof(FLTC)$	address of $FLTC$	0x69ce87af870ebefdc7dd1099ec9ea5bc 4f86208e	0x5303b5a16655C69D7914cf6fcdF5A5 429C41279F
$FLTPC.FLNFTID$	Federated Learning Non-Fungible Token ID for a FLT	1	1

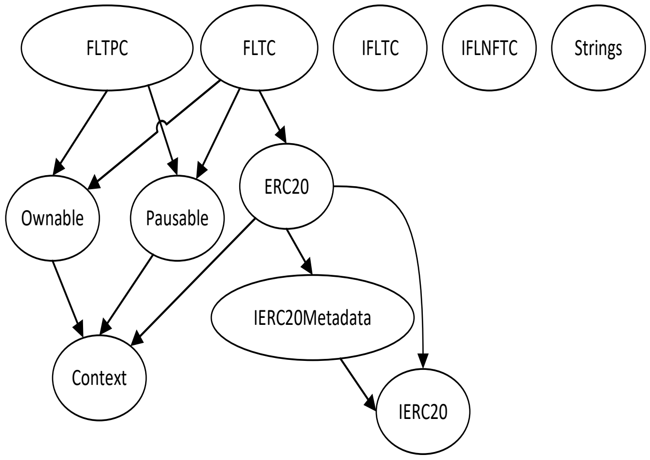


FIGURE 3: Inheritance graph of the FTPC and FLTC

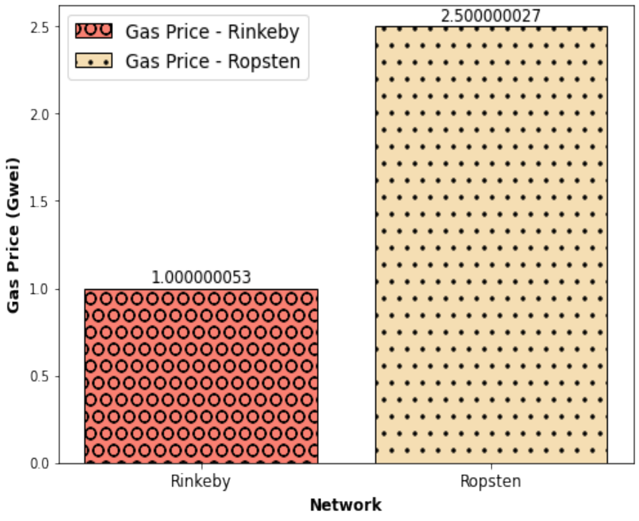


FIGURE 5: Gas Price for deployment of FLNFTC and FLTPC on Rinkeby and Ropsten

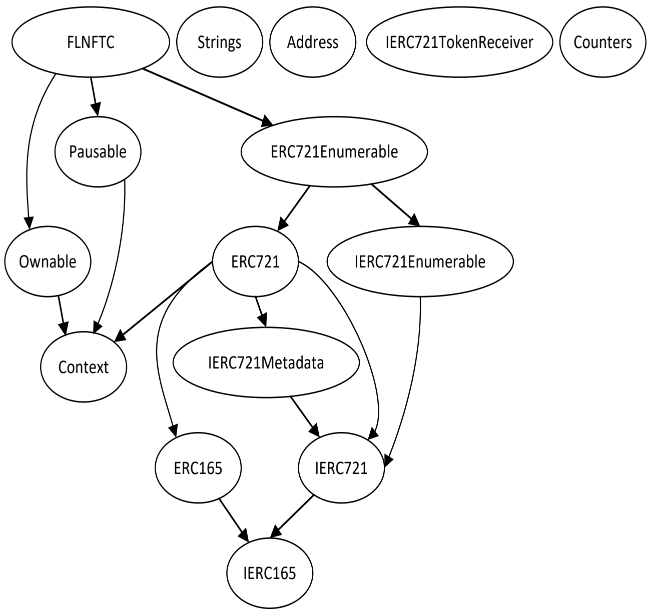


FIGURE 4: Inheritance graph of the FLNFTC

V. IMPLEMENTATION, DEPLOYMENT AND EVALUATION

This section discusses the implementation, deployment, and evaluation of the FL-Incentivizer.

A. IMPLEMENTATION AND DEPLOYMENT

We developed our contracts in Solidity. The inheritance graph of the contracts is generated using Surya [50]. The inheritance graph of FLTPC and FLTC is shown in Fig. 3. While, Fig. 4 shows inheritance graph for FLNFTC. The inheritance graphs show that the $FLTPC$ is inherited from the “Ownable contract" [51] as well as the “Pausable contract" from OpenZeppelin [52]. The FLNFTC is derived from ERC721Enumerable [53], Pausable and Ownable contracts from OpenZeppelin. Similarly, FLTC is inherited from OpenZeppelin ERC-20 implementation [54], Pausable and Ownable contracts. Fig. 13 in Appendix shows a simplified class diagram for FLTPC, FLNFTC, and FLTC.

We compiled our smart contracts using Hardhat using the command

```
npx hardhat compile
```

We deployed our smart contracts using Hardhat and JavaScript using the commands

```
 npx hardhat run script/deploy.js
--network rinkeby
```

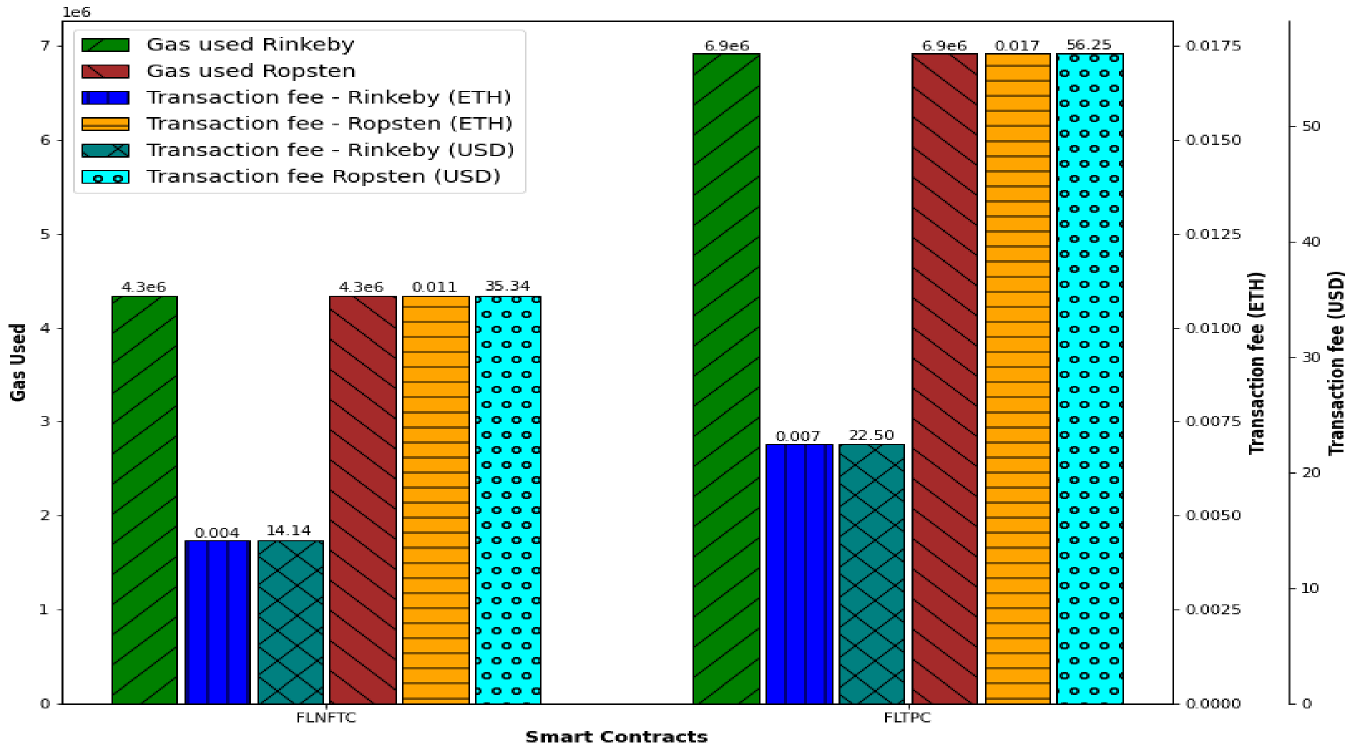


FIGURE 6: Gas used, transaction fee in ETH and USD for deployment of FLTPC and FLNFTC

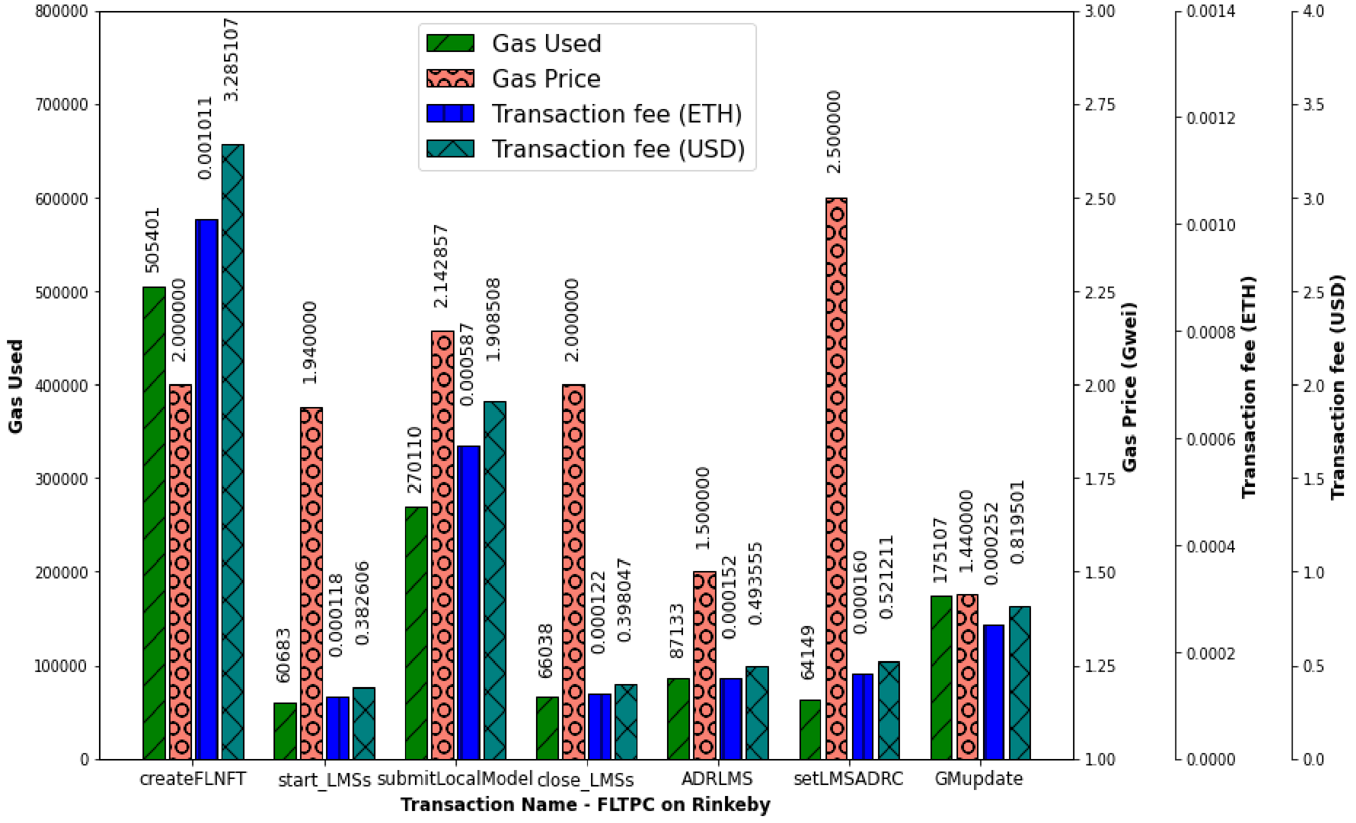
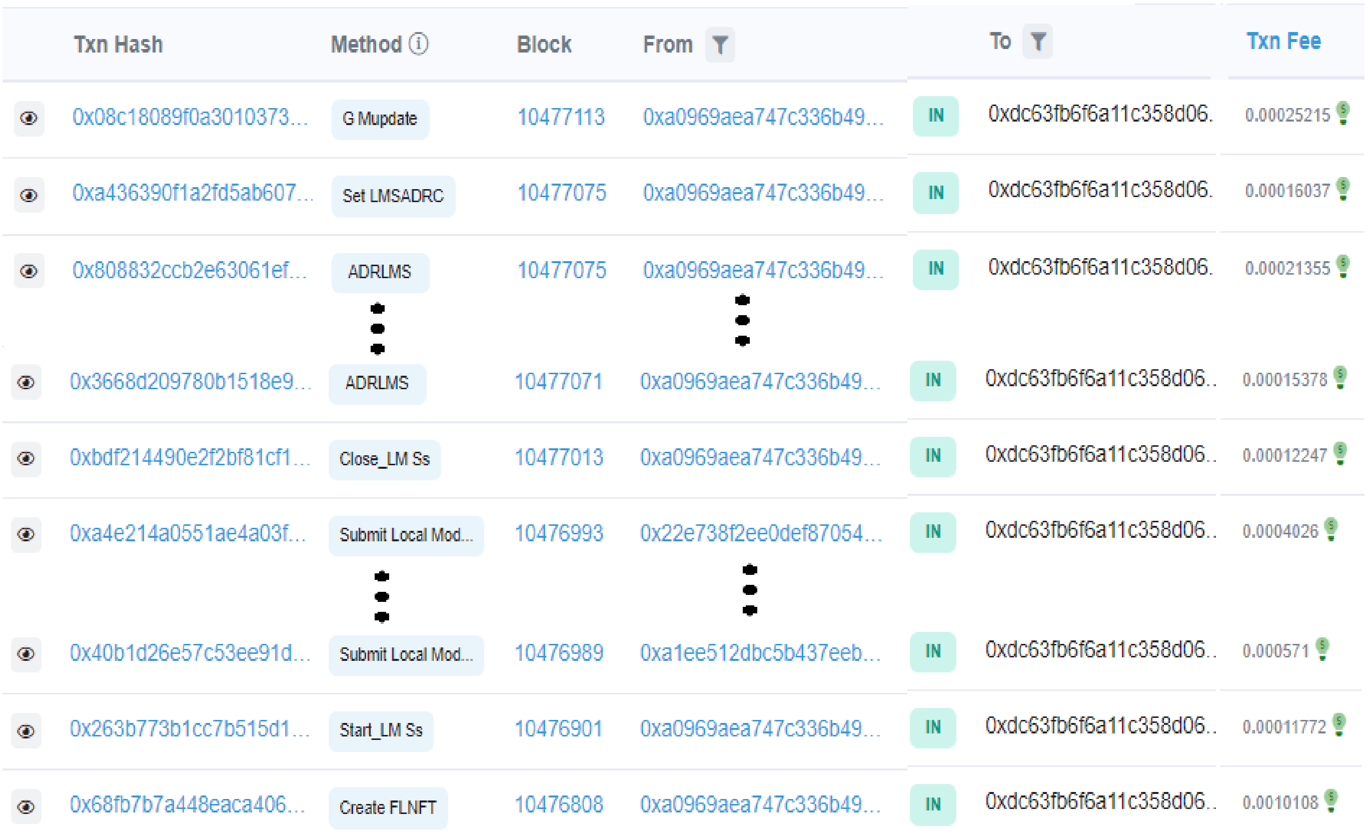
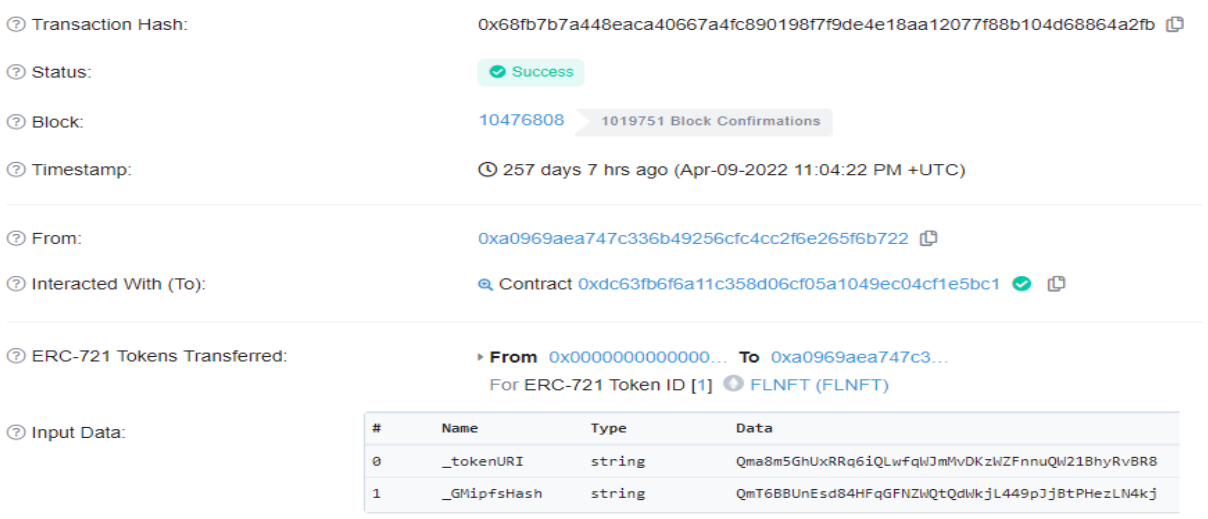


FIGURE 7: Average gas used, gas price, transaction fee (in ETH and USD) for various transaction on FLTPC



Txn Hash	Method	Block	From		To	Txn Fee
0x08c18089f0a3010373...	G Mupdate	10477113	0xa0969aea747c336b49...	IN	0xdc63fb6f6a11c358d06.	0.00025215
0xa436390f1a2fd5ab607...	Set LMSADRC	10477075	0xa0969aea747c336b49...	IN	0xdc63fb6f6a11c358d06.	0.00016037
0x808832ccb2e63061ef...	ADRLMS	10477075	0xa0969aea747c336b49...	IN	0xdc63fb6f6a11c358d06.	0.00021355
	⋮		⋮			
0x3668d209780b1518e9...	ADRLMS	10477071	0xa0969aea747c336b49...	IN	0xdc63fb6f6a11c358d06..	0.00015378
0xbdf214490e2f2bf81cf1...	Close_LM Ss	10477013	0xa0969aea747c336b49...	IN	0xdc63fb6f6a11c358d06..	0.00012247
0xa4e214a0551ae4a03f...	Submit Local Mod...	10476993	0x22e738f2ee0def87054...	IN	0xdc63fb6f6a11c358d06..	0.0004026
	⋮		⋮			
0x40b1d26e57c53ee91d...	Submit Local Mod...	10476989	0xa1ee512dbc5b437eeb...	IN	0xdc63fb6f6a11c358d06..	0.000571
0x263b773b1cc7b515d1...	Start_LM Ss	10476901	0xa0969aea747c336b49...	IN	0xdc63fb6f6a11c358d06..	0.00011772
0x68fb7b7a448eaca406...	Create FLNFT	10476808	0xa0969aea747c336b49...	IN	0xdc63fb6f6a11c358d06..	0.0010108

FIGURE 8: Transaction executed for a complete GI on FLTPC (https://rinkeby.etherscan.io/address/0xdC63FB6f6a11c358D06Cf05A1049eC04cf1e5bc1)



Transaction Hash:	0x68fb7b7a448eaca40667a4fc890198f7f9de4e18aa12077f88b104d68864a2fb
Status:	Success
Block:	10476808 1019751 Block Confirmations
Timestamp:	257 days 7 hrs ago (Apr-09-2022 11:04:22 PM +UTC)
From:	0xa0969aea747c336b49256cfc4cc2f6e265f6b722
Interacted With (To):	Contract 0xdc63fb6f6a11c358d06cf05a1049ec04cf1e5bc1
ERC-721 Tokens Transferred:	From 0x0000000000000... To 0xa0969aea747c3... For ERC-721 Token ID [1] FLNFT (FLNFT)
Input Data:	

#	Name	Type	Data
0	_tokenURI	string	Qma8m5GhUxRRq6iQLwfqWJmMvDKzWZFnnuQW21BhyRvBR8
1	_GMipfsHash	string	QmT6BBUnEsd84HFqGFNZWQtQdWkjL449pJjBtPHezLN4kj

FIGURE 9: Minting of FL-NFT to $FLTPCO$ by transaction 'Create FLNFT' (mintFLNFT) (https://rinkeby.etherscan.io/tx/0x68fb7b7a448eaca40667a4fc890198f7f9de4e18aa12077f88b104d68864a2fb)

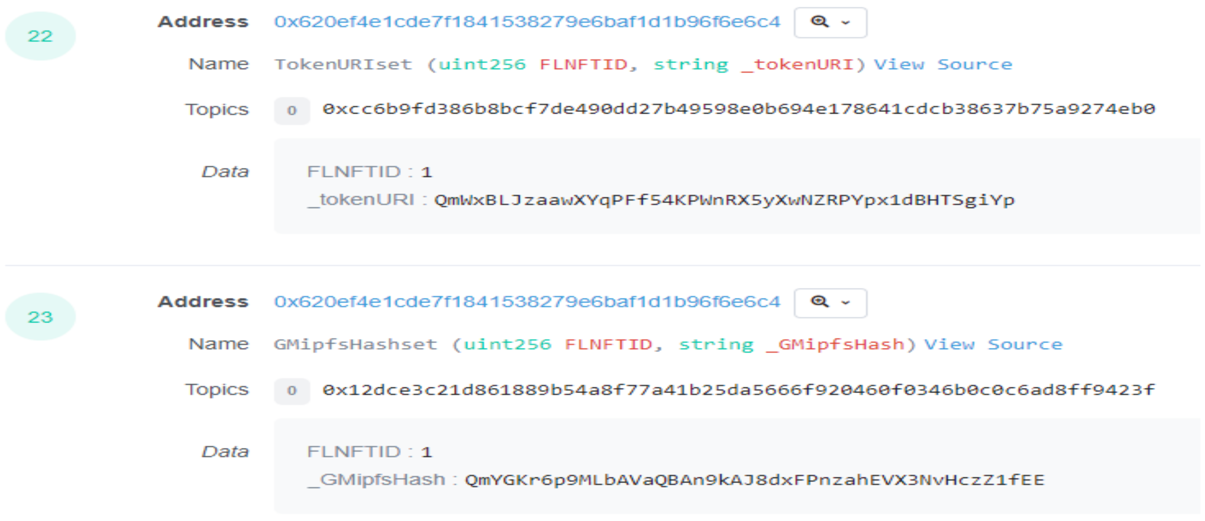
and

```
npx hardhat run script/deploy.js
--network ropsten
```

on the Rinkeby and the Ropsten network respectively. We also verified our smart contracts using

```
npx hardhat verify --network rinkeby
contract_address arg1
```

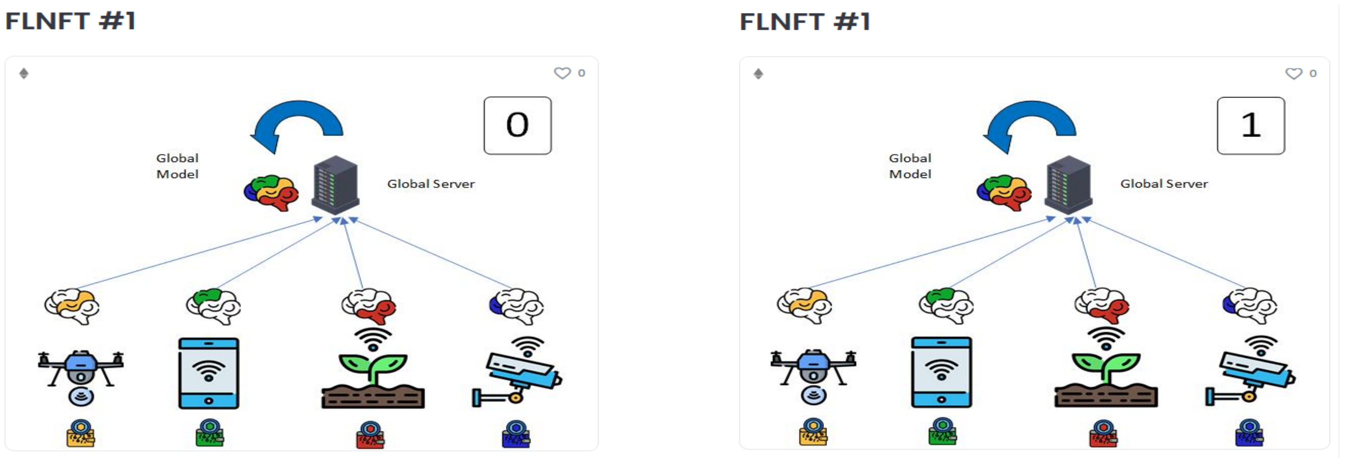
for the Rinkeby network using ETHERSCAN_API_KEY [55].



```
22    Address   0x620ef4e1cde7f1841538279e6baf1d1b96f6e6c4
      Name      TokenURIset (uint256 FLNFTID, string _tokenURI) View Source
      Topics    0  0xcc6b9fd386b8bcf7de490dd27b49598e0b694e178641cdcb38637b75a9274eb0
      Data      FLNFTID : 1
                _tokenURI : QmWxBLJzaawXYqPFf54KPWnRX5yXwNZRPYpx1dBHTSgiYp

23    Address   0x620ef4e1cde7f1841538279e6baf1d1b96f6e6c4
      Name      GMipfsHashset (uint256 FLNFTID, string _GMipfsHash) View Source
      Topics    0  0x12dce3c21d861889b54a8f77a41b25da5666f920460f0346b0c0c6ad8ff9423f
      Data      FLNFTID : 1
                _GMipfsHash : QmYGKr6p9MLbAVaQBAn9kAJ8dxFPnzahEVX3NvHczZ1fEE
```

FIGURE 10: Eventlog of transaction ‘GMupdate’ showing events TokenURIset and GMipfsHashset (https://rinkeby.etherscan.io/tx/0x08c18089f0a3010373e16259b91f5591c22deda0c0980cb4fb0d77e785240697#eventlog)



(a) for t=0 (b) for t=1

FIGURE 11: Representative image of FL-NFT on OpenSea[a].

[a]Could See FL-NFT on OpenSea at https://testnets.opensea.io/assets/0x620ef4e1cde7f1841538279e6baf1d1b96f6e6c4/1 before the Ethereum Merge. Now see property ‘image’ for URL of representative image of FL-NFT from meta-data of FL-NFT at https://ipfs.io/ipfs/QmWxBLJzaawXYqPFf54KPWnRX5yXwNZRPYpx1dBHTSgiYp

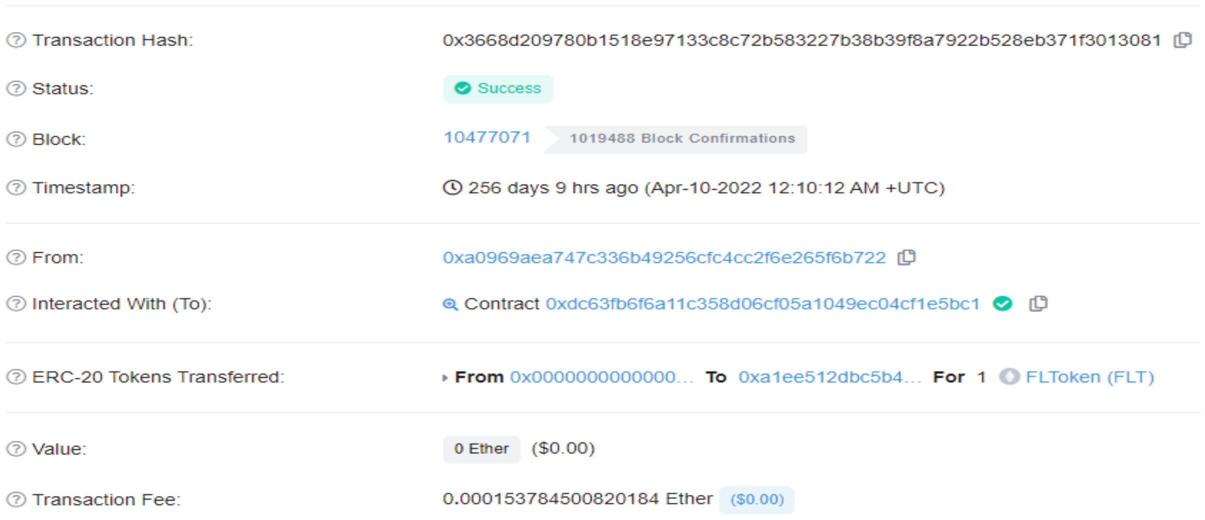
We simply used the MNIST dataset for training of the GM. As the scope of the paper is to build a generic ecosystem for incentivization of FL model training and trading using fungible tokens and NFTs. So we will skip model architecture, accuracy, and data-distribution-related details here. The EIP-170, introduced with the Spurious Dragon hard-fork [56], places a 24 KB limit on smart contracts. By optimizing our smart contracts and encoding the error messages [57], we have reduced the size of the smart contract byte-codes and, in turn, reduced the gas used for deployment and execution of smart contracts. When an externally owned account (EOA) sends some transaction to the network, it sets the gas limit for the transaction. The gas limit is the maximum gas that can be used to execute the transaction. The EOA pays a maximum transaction fee ($MaxTxFee$) for executing the transaction as determined below:

$$MaxTxFee = GasPrice * GasLimit. \tag{11}$$

The actual transaction fee depends upon the gas price and gas used for executing the transaction and can be computed as

$$Transaction\_fee = Actual\_gas\_used * Gas\_price. \tag{12}$$

The residual $MaxTxFee - Transaction\_fee$ is returned to the EAO. The transaction fee for the deployment of smart contracts in USD is calculated for 1 ETH = 3,250 USD. The



Transaction Hash:	0x3668d209780b1518e97133c8c72b583227b38b39f8a7922b528eb371f3013081
Status:	Success
Block:	10477071 1019488 Block Confirmations
Timestamp:	256 days 9 hrs ago (Apr-10-2022 12:10:12 AM +UTC)
From:	0xa0969aea747c336b49256cfc4cc2f6e265f6b722
Interacted With (To):	Contract 0xdc63fb6f6a11c358d06cf05a1049ec04cf1e5bc1
ERC-20 Tokens Transferred:	From 0x0000000000000... To 0xa1ee512dbc5b4... For 1 FLToken (FLT)
Value:	0 Ether ($0.00)
Transaction Fee:	0.000153784500820184 Ether ($0.00)

FIGURE 12: Transfer of FL-Token (symbol: FLT) to $FLTrainer_1$ for contributing a LMS for FLT (https://rinkeby.etherscan.io/tx/0x3668d209780b1518e97133c8c72b583227b38b39f8a7922b528eb371f3013081)

gas price for the deployment of FLTPC and FLNFTC on Rinkeby and Ropsten is approximately 1.000000053 Gwei and 2.500000027 Gwei respectively as shown in Fig. 5. Fig. 6 illustrates the gas consumption and transaction fee (in ETH and USD) for the deployment of FLTPC and FLNFTC. FLTPC deployed the FLTC via an internal transaction. Therefore, the gas consumption and the transaction fee for the deployment of FLTC are included in the gas used and transaction fee for the deployment of FLTPC, respectively. Fig. 6 shows that the gas used for deployment of respective smart contracts is the same on both Rinkeby and Ropsten as it depends upon the byte-code of the contract. Fig. 7 shows the average gas consumption, gas price, and transaction fee (in ETH and USD) for various transactions executed on FLTPC on Rinkeby.

Fig. 8 shows the list of transactions on FLTPC. The transaction 'Create FLNFT' (mintFLNFT) is executed by FLTPCO only once at the start. Transactions 'Start_LMSs', 'Close_LMSs', 'setLMSADRC', and 'GMupdate' are executed once per GI. Transactions 'SubmitLocalModel' and 'ADRLMS' are executed multiple times per GI according to the number of available FL-Trainers. As the transaction list is immutable, open, and transparent, this can improve the FLP flow governance and record-keeping.

Fig. 9 shows the minting of FL-NFT to $FLTPCO$ by transaction 'Create FLNFT' (mintFLNFT). The input data to the transaction include the _tokenURI and _GMipfsHash. This indicates the tokenization of the GM as NFT. Fig. 10 displays the events emitted by transaction GMupdate. Event TokenURIset indicates that the URI of FLNFT with $FLNFTID = 1$ has been updated. Similarly event GMipfsHashset indicates that the IPFS hash of FLNFT with $FLNFTID = 1$ has been updated. Fig. 11a and Fig. 11b depicts the representative image of FL-NFT for FLT on OpenSea for $t = 0$ and $t = 1$, respectively. The representative image and tokenURI of FL-NFT on OpenSea can be updated after every GI t. Fig. 12 shows a transfer of FL-Token to $FLTrainer_1$ for contributing a LMS for FLT. Fig. 14 shows the same FL-Token in a MetaMask wallet of $FLTrainer_1$. Table 2 lists the EOA address of Regulator, FLTPCO, $FLTrainer_1$ as well as the contract address of FLTPC, FLTC, FLNFTC for both Rinekeby [44] and Ropsten [45]. Etherscan Explorer for FLTPC, FLTC, and FLNFTC lists the internal and external transaction logs, verified contract code, and event log.

B. QUALITATIVE EVALUATION

1) FL-Tokens

The FL-Tokens minted in the respective FLTC by a particular FLTPC show how many learners have participated in the FLP of this FLT. The expression $FLTC.totalSupply()/10^{18}$ gives the number of FL-Tokens minted. This is a very effective way to establish the potential of FLT in the market. The FL-Tokens themselves are tradable and are an incentivization for learners.

2) FL-NFT

The FL-NFT is minted for a particular FLT. The FNFTC and corresponding FLTPC can show the complete history of the FL-NFT (for a particular tokenId=FLTPC.FLNFTID) such as tokenURI and GMipfsHash updates via the transaction log or events log. The ownership of FL-NFT is recognized as complete ownership and royalty rights of the corresponding FLT, GM, and its updates by the $Regulator$. By trading FL-NFT, these rights are transferred to the new owner.

3) FLTPC

The FLTPC can cater to continuous FLP for the FLT. The state, transaction log, and event log of FLTPC also show the complete history of the FLT, interaction with FLNFTC, and

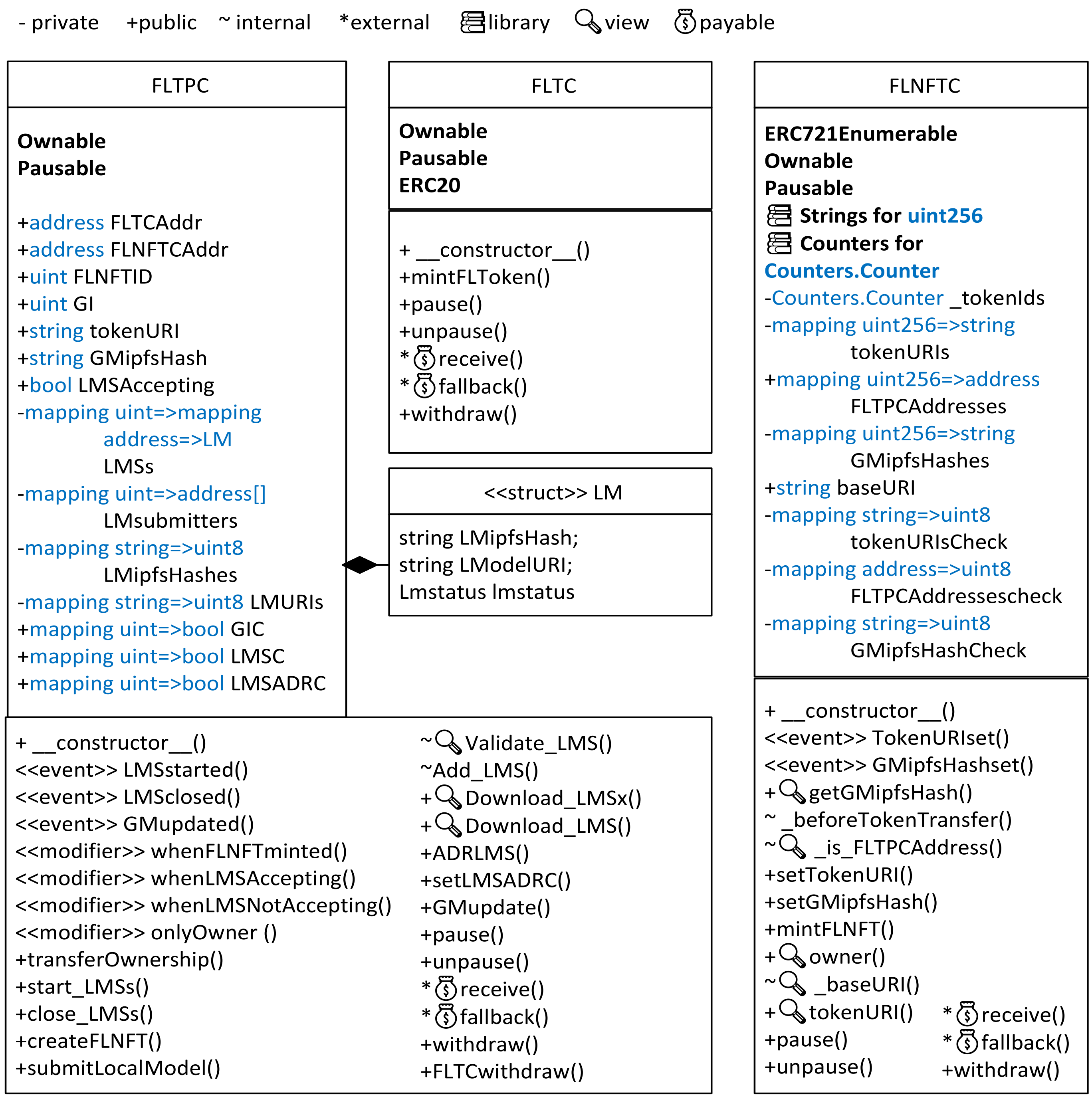


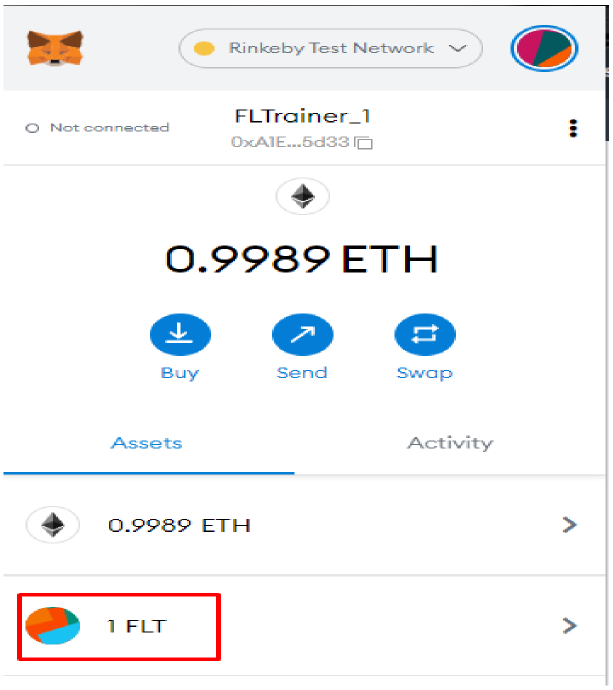
FIGURE 13: Simplified UML diagram for FLTPC, FLNFTC, and FLTC [a]

[a]Detailed UML diagram for FLTPC, FLNFTC, and FLTC at https://github.com/umermajeedkhu/FL-Incentivizer/tree/master/uml

corresponding FLTC. The trading of FL-NFT also automatically transfers ownership of the respective FLTPC to the new owner.

4) FLTPCO

The FLTPCO is responsible for minting the FL-Tokens for the partaking devices after approving their LM contributions. The FLTPCO is also responsible for correctly updating the GM using FLTPC.GMupdate. These processes rely on the honesty of FLTPCO. The best strategy for FLTPCO is to be honest, and to approve and reward only the correct LMSs. The number of FL-Tokens minted indicated how many learners were involved in the whole FLP. hus, the larger the total supply of FLTC, the higher the market value of corresponding FL-NFT. Consequently, the reliability of GM associated with FL-NFT is also increased.



0.9989 ETH

0.9989 ETH

1 FLT

FIGURE 14: MetaMask showing FL-Token (symbol: FLT) earned by $FLTrainer_1$ for contributing a LMS for FLT.

VI. CONCLUSION

The FLP requires incentives to encourage device participation. Moreover, claiming ownership of the FL global model, and demonstrating its reliability, are cumbersome processes. The paper presents a smart contract-based FL-Incentivizer framework for incentivizing the participation of devices in FLP and claiming ownership rights to a global model. As part of the FL-Incentivizer framework, transactions, logs, and the provenance of all relevant data are secured, immutable, and transparent. We present FL-NFT, a global model trading and ownership claim protocol built upon the ERC721 standard. Tradable FL-Tokens based on the ERC-20 standard are introduced to reward devices participating in FLP. FL-Incentivizer includes the InterPlanetary File System (IPFS) for storing and sharing FLP information. Detailed implementation details, smart contract code, and evaluation results are presented in this study. The number of tokens minted in FL-Token can be interpreted as a measure of the reliability of the FL global model.

.

APPENDIX A SAMPLE META-DATA OF FL-NFT

See the meta-data of FL-NFT at https://ipfs.io/ipfs/Qma8m5GhUxRRq6iQLwfqWJmMvDKzWZFnnuQW21BhyRvBR8.

APPENDIX B SMART CONTRACTS UML DIAGRAM

See Fig. 13.

REFERENCES

[1] L. U. Khan, W. Saad, D. Niyato, Z. Han, and C. S. Hong, “Digital-Twin-Enabled 6G: Vision, Architectural Trends, and Future Directions,” IEEE Communications Magazine, vol. 60, no. 1, pp. 74–80, Feb. 2022.

[2] L. U. Khan, Z. Han, W. Saad, E. Hossain, M. Guizani, and C. S. Hong, “Digital twin of wireless systems: Overview, taxonomy, challenges, and opportunities,” arXiv:2202.02559, 2022.

[3] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, “Communication-Efficient Learning of Deep Networks from Decentralized Data,” in Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, Florida, USA, Apr. 2017, pp. 1273–1282.

[4] J. Konečnỳ, H. B. McMahan, D. Ramage, and P. Richtárik, “Federated optimization: Distributed machine learning for on-device intelligence,” arXiv:1610.02527, 2016.

[5] U. Majeed and C. S. Hong, “FLchain: Federated Learning via MEC-enabled Blockchain Network,” in 20th Asia-Pacific Network Operations and Management Symposium (APNOMS), Matsue, Japan, Sep. 2019.

[6] U. Majeed, L. U. Khan, A. Yousafzai, Z. Han, B. J. Park, and C. S. Hong, “ST-BFL: A Structured Transparency Empowered Cross-Silo Federated Learning on the Blockchain Framework,” IEEE Access, vol. 9, pp. 155 634–155 650, Nov. 2021.

[7] Y. Zhan, J. Zhang, Z. Hong, L. Wu, P. Li, and S. Guo, “A Survey of Incentive Mechanism Design for Federated Learning,” IEEE Transactions on Emerging Topics in Computing, early access, Mar. 2021.

[8] U. Majeed, L. U. Khan, I. Yaqoob, S. A. Kazmi, K. Salah, and C. S. Hong, “Blockchain for IoT-based smart cities: Recent advances, requirements, and future challenges,” Journal of Network and Computer Applications, vol. 181, no. 103007, May 2021.

[9] M. Solouki and S. M. H. Bamakan, “An In-depth Insight at Digital Ownership Through Dynamic NFTs,” Procedia Computer Science, vol. 214, pp. 875–882, Nov. 2022.

[10] S. R. Pandey, N. H. Tran, M. Bennis, Y. K. Tun, Z. Han, and C. S. Hong, “Incentivize to Build: A Crowdsourcing Framework for Federated Learning,” in IEEE Global Communications Conference (GLOBECOM), Hawaii, USA, Feb. 2019.

[11] L. U. Khan, S. R. Pandey, N. H. Tran, W. Saad, Z. Han, M. N. H. Nguyen, and C. S. Hong, “Federated Learning for Edge Networks: Resource Optimization and Incentive Mechanism,” IEEE Communications Magazine, vol. 58, no. 10, pp. 88–93, Nov. 2020.

[12] T. H. Thi Le, N. H. Tran, Y. K. Tun, M. N. H. Nguyen, S. R. Pandey, Z. Han, and C. S. Hong, “An incentive mechanism for federated learning in wireless cellular networks: An auction approach,” IEEE Transactions on Wireless Communications, vol. 20, no. 8, pp. 4874–4887, Mar. 2021.

[13] M. di Angelo and G. Salzer, “Tokens, Types, and Standards: Identification and Utilization in Ethereum,” in IEEE International Conference on Decentralized Applications and Infrastructures (DAPPS), Oxford, UK, Jul. 2020.

[14] Ethereum.org, “Ethereum Improvement Proposals: ERC,” 2014, Accessed: May 15, 2022. [Online]. Available: https://eips.ethereum.org/erc

[15] Q. Wang, R. Li, Q. Wang, and S. Chen, “Non-Fungible Token (NFT): Overview, Evaluation, Opportunities and Challenges,” arXiv:2105.07447, 2021.

[16] T. Igarashi, T. Kazuhiko, Y. Kobayashi, H. Kuno, and E. Diehl, “Photrace: A Blockchain-Based Traceability System for Photographs on the Internet,” in IEEE International Conference on Blockchain (Blockchain), Melbourne, Australia, Dec. 2021, pp. 590–596.

[17] CryptoKitties.Co, “CryptoKitties: Collect and breed digital cats!” Accessed: May 15, 2022. [Online]. Available: https://www.cryptokitties.co

[18] K. B. Muthe, K. Sharma, and K. E. N. Sri, “A Blockchain Based Decentralized Computing And NFT Infrastructure For Game Networks,” in Second International Conference on Blockchain Computing and Applications (BCCA), Antalya, Turkey, Nov. 2020, pp. 73–77.

[19] A. Manzoor, M. Samarin, D. Mason, and M. Ylianttila, “Scavenger Hunt: Utilization of Blockchain and IoT for a Location-Based Game,” IEEE Access, vol. 8, pp. 204 863–204 879, Nov. 2020.

[20] V. Valaštín, K. Košt’ál, R. Bencel, and I. Kotuliak, “Blockchain Based Car-Sharing Platform,” in International Symposium ELMAR, Zadar, Croatia, Sept. 2019, pp. 5–8.

[21] P. Khezr and V. Mohan, “Property rights in the Crypto age: NFTs and the auctioning of limited edition artwork,” Available at SSRN 3900203, Aug. 2021.

[22] S. Agbesi and G. Asante, “Electronic Voting Recording System Based on Blockchain Technology,” in 2019 12th CMI Conference on Cybersecurity and Privacy (CMI), Copenhagen, Denmark, Nov. 2019.

[23] M. Stefanovic, D. Przulj, S. Ristic, D. Stefanovic, and D. Nikolic, "Smart Contract Application for Managing Land Administration System Transactions," IEEE Access, vol. 10, pp. 39 154–39 176, Apr. 2022.

[24] R. W. Ahmad, K. Salah, R. Jayaraman, I. Yaqoob, and M. Omar, "Blockchain for Waste Management in Smart Cities: A Survey," IEEE Access, vol. 9, pp. 131 520–131 541, Sept. 2021.

[25] W. Cai, Z. Wang, J. B. Ernst, Z. Hong, C. Feng, and V. C. M. Leung, "Decentralized Applications: The Blockchain-Empowered Software System," IEEE Access, vol. 6, pp. 53 019–53 033, Sept. 2018.

[26] N. Truong, G. M. Lee, K. Sun, F. Guitton, and Y. Guo, "A blockchain-based trust system for decentralised applications: When trustless needs trust," Future Generation Computer Systems, vol. 124, pp. 68–79, Nov. 2021.

[27] G. Wood et al., "Ethereum: A secure decentralised generalised transaction ledger," Ethereum project yellow paper, vol. 151, pp. 1–32, 2014.

[28] S. Wang, L. Ouyang, Y. Yuan, X. Ni, X. Han, and F. Wang, "Blockchain-Enabled Smart Contracts: Architecture, Applications, and Future Trends," IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 49, no. 11, pp. 2266–2277, Feb. 2019.

[29] Z. Zheng, S. Xie, H.-N. Dai, W. Chen, X. Chen, J. Weng, and M. Imran, "An overview on smart contracts: Challenges, advances and platforms," Future Generation Computer Systems, vol. 105, pp. 475 – 491, Apr. 2020.

[30] N. Szabo, "Smart contracts," 1994, Accessed: May 15, 2022. [Online]. Available: http://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart.contracts.html

[31] ——, "The idea of smart contracts," 1997, Accessed: May 15, 2022. [Online]. Available: http://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart_contracts_idea.html

[32] H. R. Hasan and K. Salah, "Proof of Delivery of Digital Assets Using Blockchain and Smart Contracts," IEEE Access, vol. 6, pp. 65 439–65 448, Oct. 2018.

[33] T. Hewa, M. Ylianttila, and M. Liyanage, "Survey on blockchain based smart contracts: Applications, opportunities and challenges," Journal of Network and Computer Applications, vol. 177, no. 102857, Mar. 2021.

[34] V. Buterin et al., "A next-generation smart contract and decentralized application platform, Ethereum White paper," 2014, Accessed: May 15, 2022. [Online]. Available: https://github.com/ethereum/wiki/wiki/White-Paper

[35] B. Farahani, F. Firouzi, and M. Luecking, "The convergence of IoT and distributed ledger technologies (DLT): Opportunities, challenges, and solutions," Journal of Network and Computer Applications, vol. 177, no. 102936, Mar. 2021.

[36] "MetaMask: A crypto wallet & gateway to blockchain apps," Accessed: May 15, 2022. [Online]. Available: https://metamask.io

[37] N. L. LLC, "Hardhat: Ethereum development environment for professionals," 2021, Accessed: May 15, 2022. [Online]. Available: https://hardhat.org/

[38] C. Benabbou and O. Gurcan, "A Survey of Verification, Validation and Testing Solutions for Smart Contracts," in Third International Conference on Blockchain Computing and Applications (BCCA), Tartu, Estonia, Dec. 2021, pp. 57–64.

[39] J. Chen, X. Xia, D. Lo, J. Grundy, X. Luo, and T. Chen, "Defining Smart Contract Defects on Ethereum," IEEE Transactions on Software Engineering, vol. 48, no. 1, pp. 327–345, Apr. 2022.

[40] A. Lisi, A. De Salve, P. Mori, L. Ricci, and S. Fabrizi, "Rewarding reviews with tokens: An Ethereum-based approach," Future Generation Computer Systems, vol. 120, pp. 36–54, Jul. 2021.

[41] S. Casale-Brunet, P. Ribeca, P. Doyle, and M. Mattavelli, "Networks of Ethereum Non-Fungible Tokens: A graph-based analysis of the ERC-721 ecosystem," in IEEE International Conference on Blockchain (Blockchain), Melbourne, Australia, Dec. 2021, pp. 188–195.

[42] A. Singh, K. Click, R. M. Parizi, Q. Zhang, A. Dehghantanha, and K.-K. R. Choo, "Sidechain technologies in blockchain networks: An examination and state-of-the-art review," Journal of Network and Computer Applications, vol. 149, no. 102471, Jan. 2020.

[43] H. Hamledari and M. Fischer, "The application of blockchain-based crypto assets for integrating the physical and financial supply chains in the construction & engineering industry," Automation in Construction, vol. 127, no. 103711, Jul. 2021.

[44] "Rinkeby Test Network," Accessed: May 15, 2022. [Online]. Available: https://rinkeby.etherscan.io

[45] "Ropsten Test Network," Accessed: May 15, 2022. [Online]. Available: https://ropsten.etherscan.io

[46] H. B. McMahan, E. Moore, D. Ramage, and B. A. y Arcas, "Federated learning of deep networks using model averaging," arXiv:1602.05629, 2016.

[47] U. Majeed, S. S. Hassan, and C. S. Hong, "Cross-Silo Model-Based Secure Federated Transfer Learning for Flow-Based Traffic Classification," in International Conference on Information Networking (ICOIN), Jeju Island, Korea (South), Jan. 2021, pp. 588–593.

[48] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated Learning with Non-IID Data," arXiv:1806.00582, 2018.

[49] L. U. Khan, W. Saad, Z. Han, and C. S. Hong, "Dispersed Federated Learning: Vision, Taxonomy, and Future Directions," IEEE Wireless Communications, vol. 28, no. 5, pp. 192–198, Nov. 2021.

[50] "Surya, The Sun God: A Solidity Inspector," Accessed: May 15, 2022. [Online]. Available: https://github.com/ConsenSys/surya

[51] "OpenZeppelin Ownable Implementation," Accessed: May 15, 2022. [Online]. Available: https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/access/Ownable.sol

[52] "OpenZeppelin: The premier crypto cybersecurity technology and services company," Accessed: May 15, 2022. [Online]. Available: https://openzeppelin.com

[53] "OpenZeppelin ERC721 Implementation," Accessed: May 15, 2022. [Online]. Available: https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/token/ERC721/ERC721.sol

[54] "OpenZeppelin ERC20 Implementation," Accessed: May 15, 2022. [Online]. Available: https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/token/ERC20/ERC20.sol

[55] Etherscan, "Etherscan APIs," Accessed: May 15, 2022. [Online]. Available: https://etherscan.io/apis

[56] Ethereum Foundation Blog, "Hard Fork No. 4: Spurious Dragon," Accessed: May 15, 2022. [Online]. Available: https://blog.ethereum.org/2016/11/18/hard-fork-no-4-spurious-dragon

[57] ethereum.org, "Downsizing Contracts to Fight the Contract Size Limit," Accessed: May 15, 2022. [Online]. Available: https://ethereum.org/en/developers/tutorials/downsizing-contracts-to-fight-the-contract-size-limit/

UMER MAJEED is currently pursuing his Ph.D. degree in Computer Engineering at Kyung Hee University (KHU), South Korea. He is working as a researcher in the intelligent Networking Laboratory under a project jointly funded by the prestigious Brain Korea 21st Century Plus and Ministry of Science and ICT, South Korea. He received BS degree in Electrical Engineering from the National University of Science and Technology (NUST), Pakistan in 2015. He received the best paper award in 35th IEEE International Conference on Information Networking (ICOIN), Jeju Island, South Korea, in 2021. His research interest includes Blockchain, Mobile Edge Computing, Internet of Things, ML, and Wireless Networks.

LATIF U. KHAN received his Ph.D. degree in Computer Engineering and M.S degree in Electrical Engineering with distinction from University of Engineering and Technology (UET), Peshawar, Pakistan in 2017 and Kyung Hee University (KHU), South Korea in 2021, respectively. Before joining the KHU, he has served as a faculty member and research associate in the UET, Peshawar, Pakistan. He is the recipient of KHU best thesis award 2021. He has published his works in highly reputable conferences and journals. He has received the best paper award in 15th IEEE International Conference on Advanced Communications Technology, PyeongChang, South Korea, in 2013. His research interests include federated learning, analytical techniques of optimization and game theory to edge computing and end-to-end network slicing.

SHEIKH SALMAN HASSAN (S'14) received his BS (Electrical Engineering) degree with magna cum laude from the National University of Computer and Emerging Sciences (NUCES-FAST), Karachi, Pakistan in 2017. He is currently pursuing a Ph.D. (Computer Science & Engineering) degree at Kyung Hee University (KHU), Republic of Korea. He is involved as a graduate researcher in the Networking Intelligence Laboratory under a project jointly funded by the prestigious Brain Korea 21st Century Plus and the Ministry of Science and ICT, Republic of Korea. He received the Best Poster Paper Award at the 35th International Conference on Information Networking (ICOIN) 2021 held in Jeju Island, Republic of Korea. His research interests include 6G, non-terrestrial networks, mobile edge computing, the Internet of Everything, and machine learning.

ZHU HAN (S'01, M'04, SM'09, F'14) received the B.S. degree in electronic engineering from Tsinghua University, in 1997, and the M.S. and Ph.D. degrees in electrical and computer engineering from the University of Maryland, College Park, in 1999 and 2003, respectively. From 2000 to 2002, he was an R&D Engineer of JDSU, Germantown, Maryland. From 2003 to 2006, he was a Research Associate at the University of Maryland. From 2006 to 2008, he was an assistant professor at Boise State University, Idaho. Currently, he is a John and Rebecca Moores Professor in the Electrical and Computer Engineering Department as well as in the Computer Science Department at the University of Houston, Texas. He is also a Chair professor in National Chiao Tung University, ROC. His research interests include wireless resource allocation and management, wireless communications and networking, game theory, big data analysis, security, and smart grid. Dr. Han received an NSF Career Award in 2010, the Fred W. Ellersick Prize of the IEEE Communication Society in 2011, the EURASIP Best Paper Award for the Journal on Advances in Signal Processing in 2015, IEEE Leonard G. Abraham Prize in the field of Communications Systems (best paper award in IEEE JSAC) in 2016, and several best paper awards in IEEE conferences. Dr. Han was an IEEE Communications Society Distinguished Lecturer from 2015-2018, and is AAAS fellow since 2019 and ACM distinguished Member since 2019. Dr. Han is 1% highly cited researcher since 2017 according to Web of Science.

CHOONG SEON HONG (S'95-M'97-SM'11) received the B.S. and M.S. degrees in electronic engineering from Kyung Hee University, Seoul, South Korea, in 1983 and 1985, respectively, and the Ph.D. degree from Keio University, Tokyo, Japan, in 1997. In 1988, he joined KT, Gyeonggi-do, South Korea, where he was involved in broadband networks as a member of the Technical Staff. Since 1993, he has been with Keio University. He was with the Telecommunications Network Laboratory, KT, as a Senior Member of Technical Staff and as the Director of the Networking Research Team until 1999. Since 1999, he has been a Professor with the Department of Computer Science and Engineering, Kyung Hee University. His research interests include future Internet, intelligent edge computing, network management, and network security. Dr. Hong is a member of the Association for Computing Machinery (ACM), the Institute of Electronics, Information and Communication Engineers (IEICE), the Information Processing Society of Japan (IPSJ), the Korean Institute of Information Scientists and Engineers (KIISE), the Korean Institute of Communications and Information Sciences (KICS), the Korean Information Processing Society (KIPS), and the Open Standards and ICT Association (OSIA). He has served as the General Chair, the TPC Chair/Member, or an Organizing Committee Member of international conferences, such as the Network Operations and Management Symposium (NOMS), International Symposium on Integrated Network Management (IM), Asia-Pacific Network Operations and Management Symposium (APNOMS), End-to-End Monitoring Techniques and Services (E2EMON), IEEE Consumer Communications and Networking Conference (CCNC), Assurance in Distributed Systems and Networks (ADSN), International Conference on Parallel Processing (ICPP), Data Integration and Mining (DIM), World Conference on Information Security Applications (WISA), Broadband Convergence Network (BcN), Telecommunication Information Networking Architecture (TINA), International Symposium on Applications and the Internet (SAINT), and International Conference on Information Networking (ICOIN). He was an Associate Editor of the IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT and the IEEE JOURNAL OF COMMUNICATIONS AND NETWORKS. He currently serves as an Associate Editor for the International Journal of Network Management.

• • •