DAO-FL: Enabling Decentralized Input and Output Verification in Federated Learning with Decentralized Autonomous Organizations

Umer Majeed, Sheikh Salman Hassan, *Member, IEEE*, Zhu Han *Fellow, IEEE*, and Choong Seon Hong, *Fellow, IEEE*

***Abstract*—In the rapidly evolving landscape of Web3 and blockchain technologies, decentralized autonomous organizations (DAOs) have emerged as innovative structures that operate autonomously through blockchain and smart contracts, eliminating the need for centralized control. The federated learning (FL) process, akin to an information flow under structured transparency, involves local models (LMs) as inputs and the global model (GM) as the output for each global iteration. Due to the centralized validation of LMs and GM updates, traditional FL systems lack transparency and security. To tackle these challenges, we introduce DAO-FL, a smart contract-based framework that harnesses the power of DAOs to enhance the security and transparency of FL systems. DAO-FL introduces the concept of DAO Membership Tokens (DAOMTs) as a governance tool within a DAO. DAOMTs play a crucial role within the DAO, facilitating members' enrollment and expulsion. Our framework incorporates a Validation-DAO for decentralized input verification (DIV) of the FL process, ensuring reliable and transparent validation of LMs. Additionally, DAO-FL utilizes a multi-signature approach facilitated by an Orchestrator-DAO to enable decentralized GM updates. This approach ensures decentralized output verification (DOV) of the FL process. We present a comprehensive system architecture, detailed execution workflow, implementation specifications, and qualitative evaluation for DAO-FL. Evaluation under threat models highlights DAO-FL's out-performance against traditional centralized-FL, effectively countering input and output attacks. DAO-FL excels in scenarios where DIV and DOV are crucial, offering enhanced transparency and trust. In conclusion, DAO-FL provides a compelling solution for FL, reinforcing the integrity of the FL ecosystem through decentralized decision-making and validation mechanisms.**

***Index Terms*—Decentralized autonomous organization, Decentralized input verification, Decentralized output verification, Federated Learning, DAO membership tokens, Non-transferable tokens, Smart contract, Soul-bound tokens, Structured transparency.**

I. Introduction

In the dynamic landscape of Web3 and blockchain [1] technology, several disrupting technologies have emerged, transforming the way we interact and conduct digital transactions. Decentralized autonomous organizations (DAOs) [2] represent innovative organizational structures that operate autonomously through blockchain technology and smart contracts [3], [4], eliminating the need for centralized control. DAOs have the potential to revolutionize traditional hierarchical management paradigms, reducing communication, administration, and collaboration expenses within organizations [5]. Another groundbreaking innovation is Soul-Bound tokens (SBTs) [6], [7], which are non-transferable tokens (NTTs) intrinsically linked to specific addresses, serving as unique digital identities and reputation indicators. SBTs provide enhanced security and authenticity in various applications, including identity verification and exclusive ownership rights. Furthermore, Non-fungible Tokens (NFTs) [8], [9] have emerged as a game-changer in the art and gaming industries. These tokens represent distinct and indivisible digital assets, enabling provable ownership and authenticity for digital art, collectibles, and virtual assets.

Federated learning (FL) [10]–[13] as a distributed artificial intelligence (DAI) technique facilitates the collaborative learning of a highly accurate deep learning model by aggregating local models (LMs) into a global model (GM) through the FL process. The FL process can be viewed as an information flow within the context of structured transparency (ST) [14], where LMs serve as inputs and the GM is the output for each global iteration (GI) [15]. Input and output verification are vital components in ST. Input verification (IV) validates information flow inputs, ensuring alignment with requirements. Output verification (OV) guarantees output integrity, validating policy compliance and preventing tampering. Decentralized input verification (DIV) and Decentralized output verification (DOV) distribute these verification processes across multiple entities, eliminating reliance on a single entity. In FL, IV confirms compliance of submitted LMs with process policies, while OV ensures adherence of the produced GM to process policies.

FL is a resource-intensive process that typically requires several days of training to develop an initial deployable GM and necessitates ongoing updates over extended periods. In traditional centralized FL frameworks, LMs are validated by a central server, which aggregates them to produce the GM. This reliance on a centralized server for both input and output verification creates a critical vulnerability: if the central server is compromised, a single erroneous GM update can severely undermine the accuracy and integrity of the entire FL process. Moreover, this centralized approach limits transparency and accountability, as the validation and aggregation processes are controlled by a single entity, increasing the risk of manipulation and misuse. These shortcomings highlight the urgent need for a more robust and decentralized solution that can

U. Majeed, S. S. Hassan and C. S. Hong are with the Department of Computer Science & Engineering, Kyung Hee University, Yongin-si 17104, South Korea.

Z. Han is with the Electrical & Computer Engineering Department, University of Houston, Houston, TX 77004, USA and Department of Computer Science & Engineering, Kyung Hee University, Yongin-si 17104, South Korea.

enhance the security and reliability of FL systems. To address these challenges, we propose the DAO-FL framework, which integrates DAOs and a multi-signature [16] contract with FL to facilitate Decentralized Input Verification (DIV) and Decentralized Output Verification (DOV). By leveraging the principles of decentralization, DAO-FL distributes the verification process across multiple participants, thereby enhancing transparency and significantly reducing the risks associated with central authority manipulation. This innovative approach not only strengthens the integrity of the FL process but also fosters a more collaborative and trustworthy environment for all stakeholders involved. The following is a summary of our contributions:

- We introduce DAO Membership Tokens (DAOMTs) which serve as a means for governance in systems utilizing DAOs.
- We design decentralized schemes for member enrollment and member expulsion within a DAO.
- We present a comprehensive system architecture and detailed execution workflow of DAO-FL, a framework powered by DAOs and smart contracts for partially decentralized orchestration of the FL process. The VDAO ensures DIV by validating and rewarding local model uploads (LMUs). Additionally, DAO-FL utilizes a multi-signature contract through the ODAO to ensure DOV by validating the GM updates.
- We present comprehensive implementation and deployment specifications, including the smart contract code[1]. Additionally, we provide evaluations of DAO-FL concerning threat models, qualitative assessments, and case studies. Furthermore, we discuss DAO-FL's applicability, limitations, and future direction.

The remaining sections of this article are structured as follows: Section II provides a comprehensive review of related literature on our study. In Section III, we explore the relevant preliminaries necessary for understanding our work. The system architecture and execution workflow of DAO-FL is expounded upon in Section IV. Implementation specifications, deployment details, evaluation on threat models, and qualitative evaluation of DAO-FL can be found in Section V. This section also covers the discussion on applicability, limitations, future directions of DAO-FL, and practical case studies. Finally, we conclude our paper in Section VI. Table I lists the abbreviations, symbols, and their descriptions.

II. RELATED WORK

Bluemke *et al.* in [21] examined the importance of data privacy-enhancing technologies in AI governance. They emphasized advancements in balancing privacy and performance during data exchange and analysis, highlighting the role of ST in facilitating controlled information flow. Their work addresses critical questions regarding who can access data, when, and how, ensuring efficient collaboration while minimizing the risk of data misuse.

Majeed *et al.* in [1] proposed the ST-BFL framework, integrating homomorphic encryption, FL-aggregators, FL-verifiers, and a smart contract to meet the components of ST

[1]https://github.com/umermajeedkhu/DAOFLcode/tree/main/contracts

TABLE I
LIST OF ABBREVIATIONS, SYMBOLS AND DESCRIPTION

Abbreviation	Symbol	Description
DIV	-	Decentralized input verification
DOV	-	Decentralized output verification
DAO	-	Decentralized Autonomous Organization
DAOC	-	DAO contract
DAOFLC	-	DAO-FL contract
DAOMT	-	DAO Membership Token
-	$D_{i,t+1}$	local dataset of $FLTrainer_{i,t+1}$
FLNFTC	-	FL-NFT contract
FLT	FLT	FL Task
FLTP	-	FL-task publisher
FLTokenC	-	FL-Token contract
-	$_{FLTrainer_{i,t+1}}$	i^{th} FL-Trainer for $t+1^{th}$ GI
GI	t	t^{th} Global Iteration
GM	GM	Global Model
GMCID	$GMCID$	Global Model (IPFS) Content Identifier
IPFS	-	InterPlanetary File System
IV	-	Input verification
JP	JP	Join Proposal
KP	KP	Kick Proposal
LM	LM	Local Model
LMU	$LMU_{i,t+1}$	local model upload of $FLTrainer_{i,t+1}$
LMUVDRF	-	LMU's verification, denial, and reward flag
LMURI	-	local model Uniform Resource Identifier
LMCID	-	local model (IPFS) Content Identifier
MultiSigC	-	Multi-Signature contract
-	n(DAOMT)	DAOMT supply
NFT	-	Non-fungible Token
NTT	-	Non-transferable Token
ODAO	-	Orchestrator-DAO
ODAOC	-	Orchestrator-DAO contract
ODAOM	$ODAOM_i$	i^{th} Orchestrator-DAO member
ODAOMT	-	Orchestrator-DAO Membership Token
ODAOMTC	-	Orchestrator-DAOMT contract
OV	-	Output verification
-	$Regulator$	regularity body governing FL ecosystem
SBT	-	Soul-Bound Token
ST	-	Structured Transparency
-	$tokenURI$	IPFS URI for meta data of tokens
URI	-	Uniform Resource Identifier
VDAO	-	Validation-DAO
VDAOC	-	VDAO contract
VDAOM	$VDAOM_i$	i^{th} VDAO member
VDAOMT	-	Validation-DAO Membership Token
VDAOMTC	-	Validation-DAOMT contract

TABLE II
OVERVIEW OF STRUCTURED TRANSPARENCY IN RECENT BLOCKCHAIN-ENABLED FL FRAMEWORKS AND DAO-FL

	Input Privacy	**Output Privacy**	**Input Verification**	**Output Verification**	**Flow Governance**
ST-BFL [15]	homomorphic encrypted LMs	• homomorphic encrypted GM • decryption key access restriction	-	decentralized (FL-verifiers) ($\star$ lacks detailed protocol design)	• Smart Contracts • FLTP • FL-verifiers • FL-aggregator
FL-Incentivizer [17]	basic FL privacy	basic FL privacy	centralized (FLTPCO)	centralized (FLTPCO)	• Smart Contracts • FLTPCO • FL-Tokens • FL-NFT
PureFed [18]	symmetric key encryption ($\star$ needs trusted Validator)	• symmetric key encryption • basic FL privacy	centralized (Validator)	centralized (decision controller)	• Smart Contracts • Owner • Worker & Validator • PureFed Server
OpenFL [19]	basic FL privacy	basic FL privacy	decentralized (feedback by other FL-Trainers)	- (each FL-Trainer aggregates global model locally)	• Challange - Smart Contract • Global Reputation Scores
DFL [20]	symmetric key encryption ($\star$ needs trusted blockchain nodes)	• - • basic FL privacy	decentralized (blockchain nodes) ($\star$ only tamper-based validation of local models)	decentralized ($\star$ multi-signature scheme) ($\star$ each blockchain node aggregate global model locally)	• Smart Contracts • Smart Infrastructures • Edge Servers • Blockchain Nodes
DAO-FL (This work)	basic FL privacy	basic FL privacy	decentralized (Validation-DAO) ($\star$ DAO-based voting)	decentralized (Orchestration-DAO & MultiSigC) ($\star$ DAO-based approval)	• Smart Contracts • FLTP & DAOs • FL-Tokens • FL-NFT • DAOMTs

[14] in FL process. Homomorphic encryption ensures input privacy, while FL-verifiers validate the GM for OV. However, ST-BFL does not support LM validation, as it prioritizes input privacy over IV. Moreover, details on the authentication and authorization of FL-verifiers, which are crucial for OV, are lacking. In contrast, DAO-FL emphasizes decentralized IV and OV in the FL process using DAOs. Majeed *et al.* proposed the FL-Incentivizer in [17], which incentivizes device participation in FL through FL-Tokens and provides ownership rights to a GM via FL-NFT. FL-Incentivizer employs the FLTPCO for LMs' validation and GM updates, ensuring centralized IV and OV. However, DAO-FL extends on the FL-Incentivizer by decentralizing the IV and OV processes using DAOs and a multi-signature contract.

Putra *et al.* in [18] presents PureFed, a FL framework designed to enhance efficiency, collaboration, and trustworthiness in AI model training. It leverages blockchain technology for secure, traceable interactions through smart contracts, encryption, and dual digital signatures. PureFed also implements dynamic aggregation and incentive mechanisms to optimize model convergence and promote honest collaboration. Wahrstätter *et al.* in [19] introduces OpenFL, a framework that enhances trust, scalability, and data privacy in FL through a collateral-backed reputation system. By leveraging the Ethereum blockchain and smart contracts, OpenFL requires participants to stake collateral to deter malicious behavior while emphasizing off-chain computation to reduce costs. The study demonstrates OpenFL's effectiveness with datasets like MNIST and CIFAR-10, positioning it as a promising model for future decentralized federated learning advancements.

Kalapaaking *et al.* in [20] propose an auditable and verifiable mechanism for blockchain-based decentralized federated learning (DFL), enabling scalable peer-to-peer exchange and aggregation of local models by blockchain nodes. A multi-signature scheme is used to verify the aggregated model, ensuring consensus and integrity. Edge servers run Ethereum nodes for smart-contract-based monitoring during local training, employing AES encryption to secure models before validation and aggregation by blockchain nodes. However, The article does not provide details on the access management of AES keys among blockchain nodes or the decryption process for aggregating local models. Moreover, DFL only provides tamper-based validation of local models, comparing their hashes to ensure integrity during transmission. Table. II analyzes recent blockchain-enabled FL frameworks alongside the proposed DAO-FL framework through the lens of ST components.

Lunesu *et al.* in [22] demonstrated the practical use of SBTs for COVID-19 vaccine certification through a decentralized Vaccine System DApp, leveraging blockchain. The authors highlight SBTs' potential in establishing a decentralized society and enabling self-sovereign identity (SSI). In

[23], the authors proposed using SBTs to encode individuals' affiliations and academic credentials in a decentralized network. By incorporating off-chain storage, smart contracts, and cryptographic technologies, they enhance privacy and security, offering reliable academic credential verification.

Diallo *et al.* in [24] presented an eGov-DAO system to enhance the efficiency, transparency, and security of e-government transactions. By implementing a DAO and smart contracts, the system automates transactions, reducing errors and uncertainty while ensuring accountability and mitigating corruption risks. Although the study offers a comprehensive design and potential advantages, additional research is essential to assess the practical applicability of the system in real-world government operations. Aitzhan *et al.* in [16] presented a decentralized energy trading system utilizing multi-signature transactions on the blockchain. Multi-signature ensures transaction security, requiring 2 out of 3 signatures to spend a token and preventing mediators from controlling transactions. This enhances security and protects against theft, fostering a trustworthy and competitive energy trading environment without relying on third parties.

Zhang *et al.* in [25] introduced a cross-chain digital asset system aimed at enhancing secure trading and payment by employing a relay chain alongside two parallel chains: the digital asset chain (DAC) and the payment chain (PC). This system addresses interoperability challenges between different blockchains, facilitating efficient communication and transaction security while enabling seamless digital asset exchange.

III. PRELIMINARIES

This section provides an overview of the technologies employed in the design and implementation of the DAO-FL framework.

A. Decentralized Autonomous Organization

A DAO [26] is a digital entity akin to traditional companies, empowering members to propose and vote on governance decisions typically made by boards or executives. Operating autonomously, a DAO follows business logic encoded in its smart contract to achieve its community's collective mission, supported by a token-based incentive structure. Launched in 2016, "The DAO" was the first DAO, raising $150 million in Ether (ETH) and becoming a significant digital crowdfunding project [5]. Other notable DAOs include DigixDAO, Aragon, and Steemit. The creation of a DAO starts with Externally Owned Accounts (EOAs) [1] sending Ether to the DAO's smart contract address, after which DAO tokens are issued to these EOAs, signifying membership and voting rights. DAOs facilitate various objectives, such as delivering services, raising funds, managing smart assets, coordinating with other autonomous software, and fostering stakeholder collaboration.

B. Structured Transparency

Structured transparency [14] is a framework that balances privacy and transparency in information flows, consisting of five components: input privacy, output privacy, IV, OV, and flow governance. Input privacy ensures that confidential information is processed without exposure to unauthorized parties, while output privacy allows data contribution without the risk of revealing sensitive input. IV involves verifying the integrity and authenticity of input, whereas OV ensures that the output remains untampered. Flow governance manages and controls the overall information flow. Each component has specific requirements: input privacy requires secure processing mechanisms; output privacy demands protection against the inference of sensitive input data from the output; IV needs methods for input integrity verification; OV requires techniques to confirm output integrity; and flow governance necessitates comprehensive control over the entire information flow.

C. Multi-signature wallet

A multi-signature (multisig) wallet is a digital wallet that enhances security by requiring multiple approvals before a transaction is executed [27]. Transactions in multisig wallets are governed by the quorum quotient (m-of-n ratio), which specifies the minimum number of signatories needed to authorize a transaction, relative to the total number of signatories [16]. For example, a 3-of-5 wallet requires at least three of five signers to approve a transaction. This setup is beneficial for scenarios involving multiple parties that must agree on a transaction or when additional security is needed to prevent unauthorized activities. Multisig wallets are widely used in various domains, such as financial transactions, corporate governance, and cryptocurrency exchange management. They often leverage smart contracts to enforce the requirement of multiple signatures for transaction approval, ensuring robust security and consensus before execution.

IV. PROPOSED FRAMEWORK

This section provides a detailed explanation of the proposed system architecture and execution workflow within the DAO-FL framework. As illustrated in Fig. 1, the system architecture consists of three main components: the administrative block, the decentralized block, and the FL-trainer block.

The administrative block in the DAO-FL framework includes key stakeholders such as the regulator, FL-task-publisher (FLTP), Orchestrator-DAO (ODAO), and Validation-DAO (VDAO). These entities manage and coordinate various functions within the DAO-FL ecosystem. The regulator, denoted as $Regulator$ oversees the FL ecosystem, deploys the FLNFTC, and standardizes FL-NFTs metadata. When an FLTP entity uses the DAO-FL framework to train an FL model, it must deploy specific smart contracts: ODAOC, VDAOC, DAOFLC, and MultiSigC, tailored for the FL task (FLT). The ODAO, overseeing the FL process, comprises multiple members ($ODAOM_i$). Orchestrator-DAO members (ODAOMs) approve FLTP's proposals and possess the ability to aggregate LMs. The VDAO verifies LMs submitted by FL-Trainers through its VDOA-members (VDAOMs), where each $VDAOM_i$ validates LMs pertinent to the FLT.

The decentralized block comprises several key components: FL-NFT contract (FLNFTC), ODAO contract (ODAOC), Orchestrator-DAOMT contract (ODAOMTC), VDAO contract (VDAOC), Validation-DAOMT contract (VDAOMTC), DAO-FL contract (DAOFLC), Multi-Signature contract (MultiSigC),

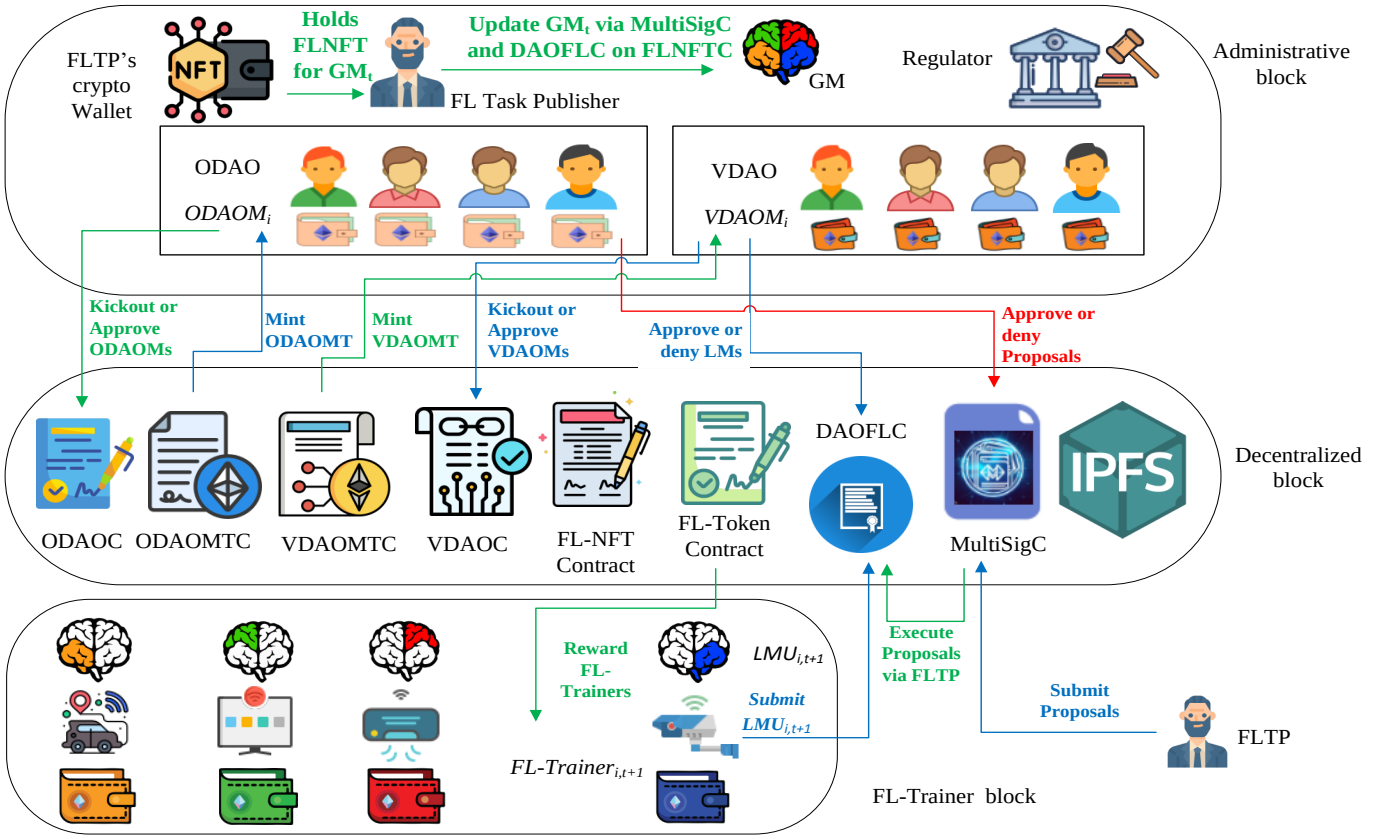

Fig. 1. DAO-FL: System Architecture.

FL-Token contract (FLTokenC), and InterPlanetary File System (IPFS). FLNFTC, based on ERC-721 and deployed by the regulator, tokenizes FLT's GM. ODAOC and VDAOC manage memberships for ODAO and VDAO, respectively. ODAOMTC and VDAOMTC mint Orchestrator-DAOMTs (ODAOMTs) and Validation-DAOMTs (VDAOMTs) for their corresponding members. Both contracts are ERC-721 customizations deployed by their respective DAO contracts. A detailed explanation of DAOMTs is available in Section IV-A. DAOFLC orchestrates the FL process for the FLT, supported by MultiSigC for decentralized verification through collective signatures from $ODAOM_i$. FLTokenC, deployed by DAOFLC, manages FL-Tokens for FLT. IPFS [17] provides decentralized storage for metadata, LMs, and GM.

The FL-Trainers block comprises multiple FL learners, where each FL-Trainer represents a device participating in the FL process. The FL-Trainer for the i^{th} device in the $t+1^{th}$ GI of an FLT is denoted as $FLTrainer_{i,t+1}$. Each FL-Trainer retrieves and downloads the GM_t, then generates its local model upload $LMU_{i,t+1}$ using its local dataset $D_{i,t+1}$.

The system architecture also includes two other key components: FL-NFTs and FL-Tokens. Each FL-NFT, denoted as $FLNFT$ and identified by a unique id $FLNFTID$, is a dynamic ERC-721 compliant NFT associated with a specific FLT. It includes an IPFS-based Uniform Resource Identifier (URI), $tokenURI$, linking to the metadata of the current GM [17], along with a $GMCID$ representing the IPFS Content Identifier (CID) [17] of the latest GM. Additionally, each FL-NFT holds the address of its DAOFLC, defined internally as $OrchestratorAddress$. The $tokenURI$, $GMCID$, and $OrchestratorAddress$ for each FL-NFT are distinctive. The FLTP, as the FL-NFT owner, manages GM commercialization and tokenization. FL-Tokens, symbolized as $FLToken$, adhere to the ERC-20 standard and are awarded to FL-Trainers for their participation in the FL process [17].

The subsequent subsections are outlined as follows: Section IV-A introduces the concept of DAOMTs. Section IV-B proposes a member enrollment scheme for adding new DAO members, while Section IV-C presents a scheme for expelling inactive or malicious members. Section IV-D outlines a mechanism for transferring ODAOC or VDAOC ownership. In Section IV-E, we propose a partially decentralized orchestration scheme for the FL process in the DAOFLC using MultiSigC. Section IV-F details the DAO-FL framework's execution workflow, orchestrating the FL process from initial setup to completing a full GI. Section IV-G explores GM commercialization through the transfer of FL-NFT and smart contracts' ownership.

A. DAO Membership Tokens (DAOMTs)

DAOs are decentralized organizations governed by members through a voting-based decision process. DAOMTs are tokens specifically designed to represent membership within a DAO. Classified as NTTs and SBTs, they cannot be traded or transferred [6]. As unique NFTs, DAOMTs are created (minted) or destroyed (burned) to manage membership. Typically, members hold only one token per address. DAOMTs can also be grouped into collections to represent different membership levels or types. They play a vital role in DAO governance, granting members voting rights on proposals and

Algorithm 1 : Membership Enrollment via DAOC

Caller: $DAOM_p$ **Output:** ***new*** JP
procedure proposeJoin(**address** $candidate$)
  Ensure $DAOM_p \in DAO$
  if $candidate \notin DAO$ and $!JoinProposals[candidate].open$ **then**
    Create ***new*** $JP \leftarrow$ {$proposer$: $DAOM_p$, $candidate$: $candidate$, $open$: true, $approvalvotes$: 0, $denialvotes$: 0, $voters$: $\emptyset$}
    Add JP to $JoinProposals$
  else
    Revert
  end if
end procedure
Caller: $DAOM_v$ **Output:** ***updated*** JP, [DOAMT **Minted** for $candidate$]
procedure voteJoin(**address** $candidate$, **bool** V_v)
  Ensure $DAOM_v \in DAO$
  $JP \leftarrow JoinProposals[candidate]$
  if $JP.open$ and $DAOM_v \notin JP.voters$ **then**
    if V_v==true **then**
      Add Approval vote for $DAOM_v$ in JP
    else
      Add Deny vote for $DAOM_v$ in JP
    end if
    Count $JP.approvalvotes$ and $JP.denialvotes$
    Q = 60% $*$ $n(DAOMT)$
    if $JP.approvalvotes > Q$ **then**
      Mint DOAMT for $candidate$
    end if
    if $JP.approvalvotes > Q$ or $JP.denialvotes > Q$ **then**
      Set $JP.open = false$
    end if
  else
    Revert
  end if
end procedure

facilitating decentralized decision-making. By representing the membership, DAOMTs help ensure a democratic approach to managing and guiding the DAO's operations.

B. Membership Enrollment in ODAO and VDAO

The process of joining ODAO or VDAO follows a similar approach. This section outlines the steps for joining a DAO through a DAO contract (DAOC), which is inherited by both ODAOC and VDAOC. After DAO creation, pre-existing members denoted as $DAOM_i \in DAO$, are required. The sequential steps for joining a DAO are summarized below:

- Step 1: When a new $candidate$ seeks to join the DAO, a current member ($DAOM_p$) initiates a "proposeJoin" transaction to the DAOC, providing the $candidate$'s address to propose their inclusion in the DAO.
- Step 2: The DAOC validates that $DAOM_p$ holds a DAOMT.
- Step 3: If the candidate is not a current DAO member and no "Join Proposal" (JP) exists for the candidate's address, a new JP is created. Proposed by $DAOM_p$, the JP is marked "open" for processing, with $approvalvotes$ and $denialvotes$ set to 0. The voter set is empty, indicating no votes have been cast yet.
- Step 4: The JP is stored in the $JoinProposals$ mapping, indexed by the $candidate$'s address.

Steps 1-4 are combined in the procedure $proposeJoin$ (Algo. 1). DAO members vote to accept or reject the JP as follows:

- Step 5: When a $DAOM_v$ votes on a JP, they initiate a "voteJoin" transaction in DAOC, submitting the

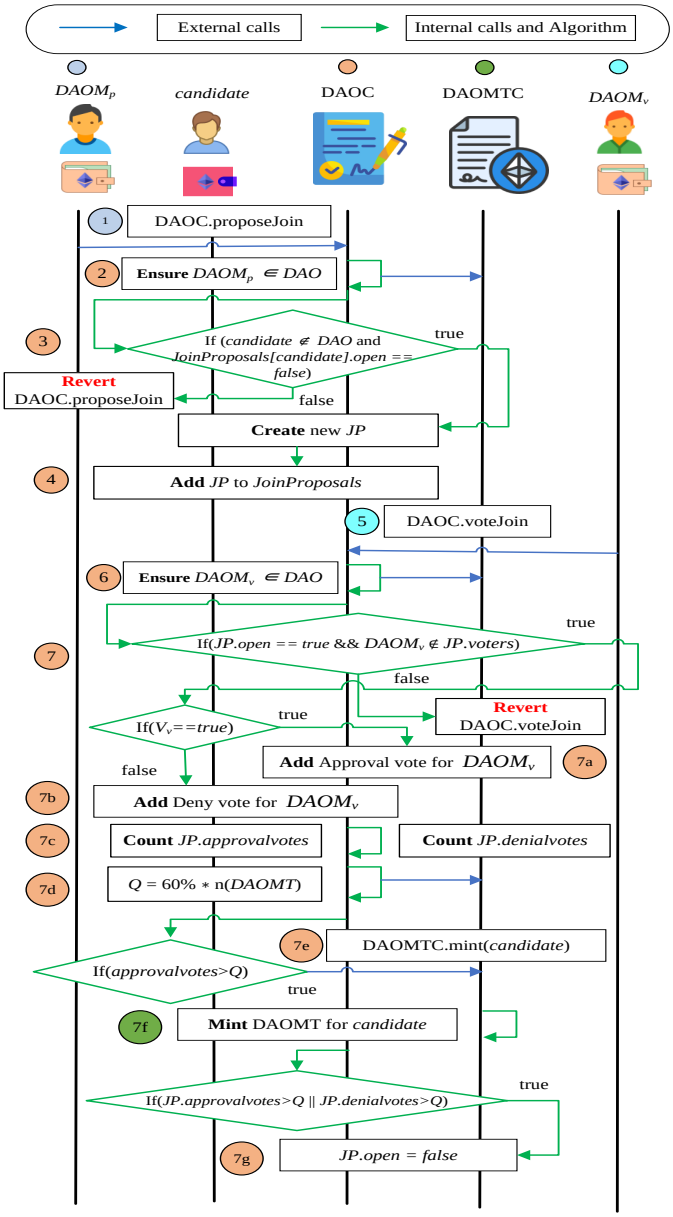

Fig. 2. Membership Enrollment in DAO - Sequence diagram.

$candidate$'s address and a boolean variable (V_v) representing their decision. A "true" value indicates approval, while "false" signifies disapproval.
- Step 6: To prevent spam, the DAOC verifies that $DAOM_v$ holds a DAOMT.
- Step 7: If an open JP exists for the $candidate$ and $DAOM_v$ hasn't voted yet, their vote is added to $JP.voters$. The total number of approval and denial votes are tallied as:

$$JP_{approvalvotes} = \sum_{V_v \in JP.voters} \mathbf{1}_{V_v==\text{true}}, \quad (1)$$

and

$$JP_{denialvotes} = \sum_{V_v \in JP.voters} \mathbf{1}_{V_v==\text{false}} \quad (2)$$

respectively. The quorum is $Q = 60\% \times n(DAOMT)$, where $n(DAOMT)$ is the DAOMTC's total supply. If $JP_{approvalvotes}$ exceeds Q, DAOC mints a DAOMT for the candidate via DAOMTC and closes the JP by setting

Algorithm 2 : Member Expulsion via DAOC

Caller: $DAOM_p$ **Output:** ***new*** KP
procedure proposeKick(**address** $candidate$)
  Ensure $DAOM_p \in DAO$
  if $candidate \in DAO$ and $!KickProposals[candidate].open$ **then**
    Create ***new*** $KP \leftarrow \{proposer$: $DAOM_p$, $candidate$: $candidate$, $open$: true, $approvalvotes$: 0, $denialvotes$: 0, $voters$: $\emptyset\}$
    Add KP to $KickProposals$
  else
    Revert
  end if
end procedure
Caller: $DAOM_v$ **Output:** ***updated*** KP, [DOAMT **Burned** for $candidate$]
procedure voteKick(**address** $candidate$, **bool** V_v)
  Ensure $DAOM_v \in DAO$
  Ensure $candidate \in DAO$
  $KP \leftarrow KickProposals[candidate]$
  if $KP.open$ and $DAOM_v \notin KP.voters$ **then**
    if V_v==true **then**
      Add Approval vote for $DAOM_v$ in KP
    else
      Add Deny vote for $DAOM_v$ in KP
    end if
    Count $KP.approvalvotes$ and $KP.denialvotes$
    Q = 60% $* n(DAOMT)$
    if $KP.approvalvotes > Q$ **then**
      Burn DOAMT owned by $candidate$
    end if
    if $KP.approvalvotes > Q$ or $KP.denialvotes > Q$ **then**
      Set $KP.open = false$
    end if
  else
    Revert
  end if
end procedure

the "open" flag to false. If $JP_{denialvotes}$ exceeds Q, the JP is rejected, and the "open" flag is set to false.

Steps 5-7 are consolidated in the procedure $voteJoin$ (Algo. 1). Fig. 2 visually illustrates the process of joining a DAO.

C. Member Expulsion in ODAO and VDAO

The presence of inactive or malicious members in a DAO necessitates their removal. Inactive members fail to participate in the FL process, while malicious members endorse inaccurate updates. Both ODAO and VDAO use the same kick-out mechanism for expelling such members. The mechanism follows these sequential steps:

- Step 1: When a DAO member ($DAOM_p$) identifies another member ($candidate$) for expulsion, $DAOM_p$ begins the kick-out process by submitting a "proposeKick" transaction to the DAOC, including the $candidate$'s address as an argument.
- Step 2: DAOC confirms that $DAOM_p$ possesses a $DAOMT$, preventing spam transactions.
- Step 3: If the $candidate$ is a DAO member and no existing "Kick Proposal" (KP) is underway for them, a new KP is initiated. The $candidate$ becomes the target, and $DAOM_p$ serves as the proposer. The KP is marked "open," indicating it is pending a decision. Both $approvalvotes$ and $denialvotes$ are set to zero, and the $KP.voters$ list is empty.
- Step 4: The KP is added to the $KickProposals$ mapping, indexed by the $candidate$'s address.

Steps 1-4 are consolidated into the procedure $proposeKick$ (Algo. 2). The voting process, carried out by DAO members for the KP, includes the following steps:

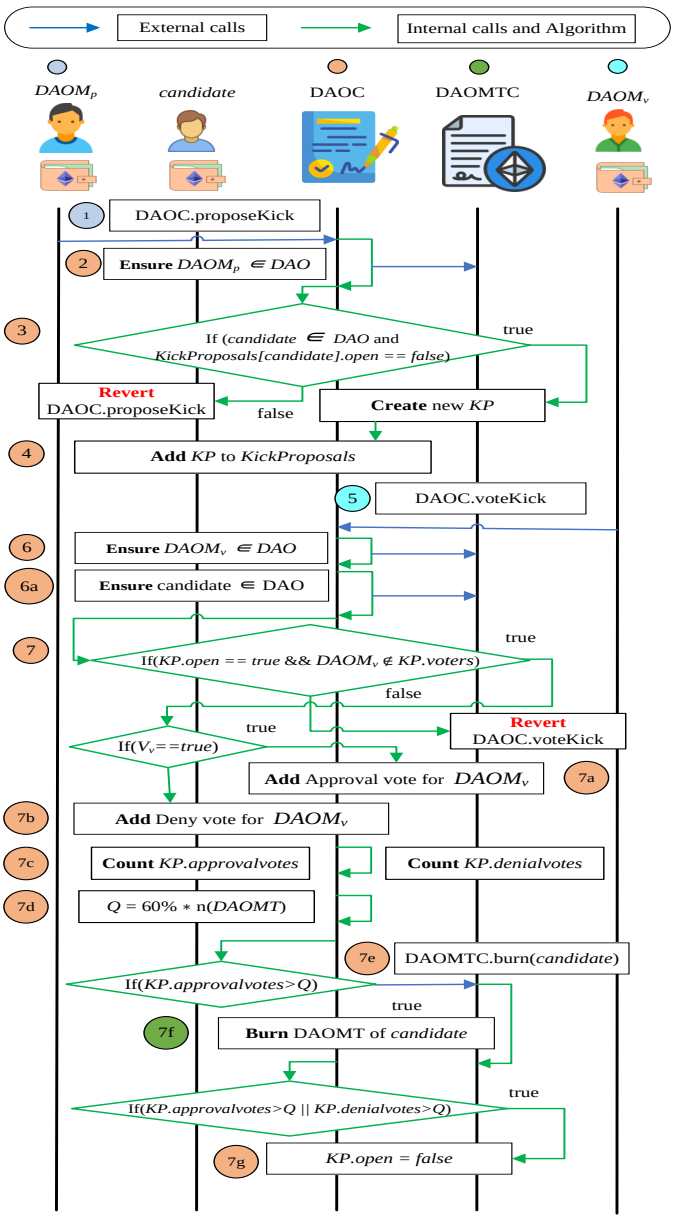

Fig. 3. Member Expulsion from DAO -Sequence diagram.

- Step 5: In the DAO's kick proposal voting process, a member ($DAOM_v$) can vote via a "voteKick" transaction to DAOC. This transaction includes the $candidate$'s address and a boolean (V_v) indicating approval (true) or disapproval (false).
- Step 6: DAOC verifies that both $DAOM_v$ and the $candidate$ possess a DAOMT.
- Step 7: If the KP is "open" for the $candidate$ and $DAOM_v$ has not yet voted, their vote is recorded in $KP.voters$. The total approval and denial votes are counted as:

$$KP_{approvalvotes} = \sum_{V_v \in KP.voters} \mathbf{1}_{V_v==\text{true}}, \quad (3)$$

and

$$KP_{denialvotes} = \sum_{V_v \in KP.voters} \mathbf{1}_{V_v==\text{false}} \quad (4)$$

Algorithm 3 : Transferring $DAOC$

Caller: FLTP **Modifier:** onlyOwner() **Output:** [Ownership ***transferred***]
procedure transferOwnership(**address** $newOwner$)
  $oldOwner \leftarrow$ owner()
  if $oldOwner \neq newOwner$ **then**
    Transfer DOAC to $newOwner$
    if $newOwner \notin DAO$ **then**
      Mint a DAOMT for $newOwner$
      Burn the DAOMT of $oldOwner$
    end if
  end if
end procedure

respectively. The quorum is $Q = 60\% \times n(DAOMT)$, where $n(DAOMT)$ is the DAOMTC's total supply. If the $KP_{approvalvotes}$ exceed the threshold Q, DAOC proceeds to burn the DAOMT held by the $candidate$, thereby closing the KP by setting the "open" flag to false. Conversely, if $KP_{denialvotes}$ exceed Q, the KP is rejected, and the "open" flag is likewise set to false.

Steps 5-7 for a KP are summarized in the procedure $voteKick$ (Algo. 2). The sequential process for expelling a member from a DAO is illustrated in Fig. 3.

D. Transferring ODAOC and VDAOC

The FLTP owns the GM, verified by the FL-NFT in FLNFTC. The FLTP also owns ODAOC and VDAOC. Upon transferring the FL-NFT ownership, ODAOC and VDAOC must also be transferred. The steps for transferring DAOC ownership, outlined in the procedure $transferOwnership$ (Algo. 3), are as follows:

- Step 1: The current owner (FLTP) initiates a "transfer ownership" transaction to DAOC, specifying the new owner's ($newOwner$) address.
- Step 2: DAOC confirms that $newOwner$ is distinct from the current owner and completes the ownership transfer. If $newOwner$ is not a DAO member, a DAOMT is minted for them, and the $oldOwner$'s DAOMT is burned to maintain scarcity.

E. Partially Decentralized Orchestration of FL process in DAOFLC through Multi-Signature Contract

In a multi-signature wallet setup, the Multi-Signature Contract (MultiSigC) gathers signatures or votes from specified individuals. Once the quorum is met, MultiSigC executes the transaction within the designated contract. In DAO-FL, MultiSigC collects votes from ODAOMs for decentralized approval of proposals, enabling transaction execution in the DAOFLC to orchestrate the FL process. Although MultiSigC manages decentralized approval, the FLTP retains sole responsibility for executing approved proposals, resulting in partial decentralization. This sequential process, illustrated in Fig. 4 is as follows:

- Step 1: The FLTP initiates a transaction "propose" (or "proposecreateFLNFT" or "proposeUpdateGM") with specific arguments submitted to MultiSigC. This transaction can address proposals such as "createFLNFT," "Initiate_LMUs," "Cease_LMUs," "setLMUVDRF," or "UpdateGM." After verifying the transaction's origin, MultiSigC validates it against the arguments, proposal type, and current state. Once validated, a new $proposal$ is created with a unique $proposalID$ and is set to "Open" state. The $selector$ of $proposal$ is configured using the corresponding function signatures [28] in DAOFLC. The FLTP then seeks ODAOMs' approval off-chain. This step is outlined in the procedure $propose$ (Algo. 4).
- Step 2: ODAOMs evaluate the proposal off-chain, considering its attributes, nature, and the states of MultiSigC and DAOFLC. If valid, an $ODAOM_v$ submits an "approve" transaction with the $proposalID$ to MultiSigC. MultiSigC verifies that the transaction is legitimate, appropriate, and timely. It also ensures the $proposal$ is open and that $ODAOM_v$ hasn't previously voted. Upon successful validation, a approval vote is recorded. The cumulative approvals are defined as:

$$num_{Approvals} = \sum_{ODAOM_v \in \text{proposal.approvals}} 1. \quad (5)$$

If the cumulative approvals exceed the quorum Q (60% of ODAMTC supply), the $proposal$'s state is updated to the "Executable". This step is outlined in procedure $approve$ (Algo. 4).

Algorithm 4 : MultiSigC

Caller: $FLTP$ **Modifier:** onlyOwner()
Output: [new $Proposal$ ($proposal$)]
procedure propose([$selector$], [$tokenURI$], [$GMCID$], [$t+1$])
  Require:*Caller==MultiSigC.owner()*
  if **Validate**($propose$) **then**
    $proposal \leftarrow$ **Create** new $Proposal$ **with** $proposalID$
    Set *proposal.state = "Open", proposal.selector*
  else
    Revert
  end if
end procedure
Output: [$proposal$ ***executed***]
procedure execute(**uint** $proposalID$)
  Require:*Caller==MultiSigC.owner()*
  if *Proposal[proposalID].state ==*"Executable" **then**
    selector = Proposal[proposalID].selector
    argumentData = Proposal[proposalID].argumentData
    if Call DAOFLC.selector **with** *argumentData* **then**
      Set *proposal.state = "Executed"*; **Update** *state* of MultiSigC
    end if
  else
    Revert
  end if
end procedure
Output: [$proposal$ ***closed***]
procedure closeProposal(**uint** $proposalID$)
  Require:*Caller==MultiSigC.owner()*; *state ← Proposal[proposalID].state*
  if *state == Open* **or** *state == "Executable"* **then**
    Set *Proposal[proposalID].state = "Closed"*
  else
    Revert
  end if
end procedure

Caller: $ODAOM_v$ **Output:** ***updated*** $proposal$, [$proposal$ ***executable***]
procedure approve(**uint** $proposalID$)
  Ensure $ODAOM_v \in ODAO$; *proposal ← Proposal[proposalID]*
  if *proposal.state == "Open"* and $ODAOM_v \notin$ *proposal.approvals* **then**
    Add $ODAOM_v$ to *proposal.approvals*
    numApprovals = proposal.approvals.length()
    if *numApprovals* $>$ (*Q* = $60\% * n(ODAOMT)$) **then**
      Set *proposal.state = "Executable"*
    end if
  else
    Revert
  end if
end procedure

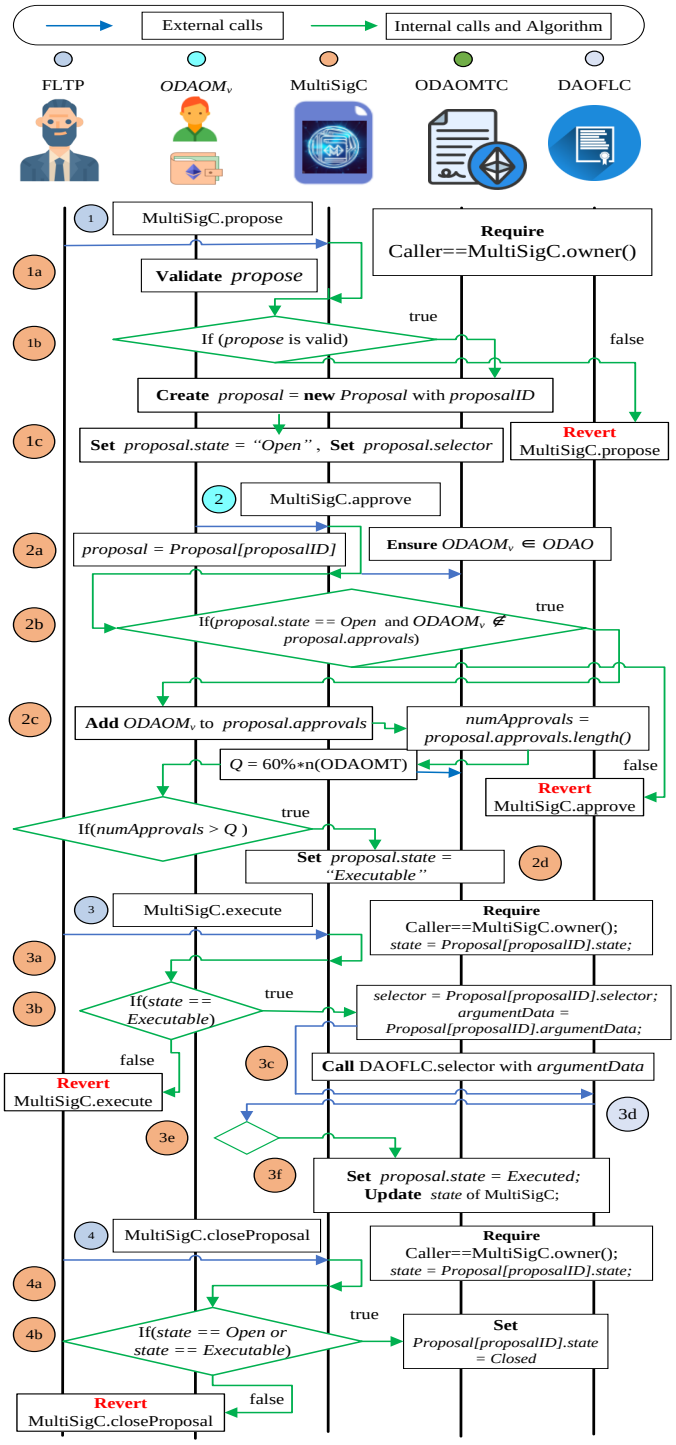

Fig. 4. Partially Decentralized Orchestration of FL process in DAOFLC through MultiSigC - Sequence diagram.

- Step 3: Once a *proposal* has the required approvals, FLTP initiates its execution by submitting an "execute" transaction with the $proposalID$ to MultiSigC. MultiSigC verifies the *proposal*'s executability based on its state and the current MultiSigC state. If valid, the proposal is executed within DAOFLC, and its state is updated. This process is detailed in the procedure $execute$ (Algo. 4). FLTP then submits the next "propose" transaction to maintain DAO-FL operations.

Algorithm 5 : FLNFTC **Owner & Deployer:** *Regulator*

Input: "Federated Learning NFT", "FLNFT", *base_URI*
Output: FLNFTC ***deployed***
procedure FLNFTC_Constructor(*_name*, *_symbol*, *base_URI*)
    $FLNFTC \leftarrow \{owner: Regulator.address, name: \_name, symbol: \_symbol, base\_URI: base\_URI\}$
end procedure
Output: FLNFT ***minted*** **Executor:** DAOFLC
procedure craftFLNFT($GMCID$, $tokenURI$)
    $FLNFTID$ = **Mint** $FLNFT$ transferred to $FLTP$
    $FLNFT \leftarrow \{tokenURI: tokenURI, GMCID: GMCID, OrchestratorAddress: DAOFLC.address\}$
end procedure
Output: [FLNFT ***updated***] **Executor:** DAOFLC
procedure assignGMCID($GMCID$, $FLNFTID$)
    if FLNFTC.Verify_GMCID($FLNFTID$, $GMCID$) **then**
        Assign $GMCIDs[FLNFTID] \leftarrow GMCID$
        Ensure Distinct $GMCIDs$
        Emit $GMCIDset(FLNFTID, GMCID)$
        Return true
    end if
end procedure
Output: [FLNFT ***updated***] **Executor:** DAOFLC
procedure assignTokenURI($tokenURI$, $FLNFTID$)
    if FLNFTC.Verify_TokenURI($tokenURI$, $FLNFTID$) **then**
        Assign $tokenURIs[FLNFTID] \leftarrow tokenURI$
        Ensure Distinct $tokenURIs$
        Emit $TokenURIset(FLNFTID, tokenURI)$
        Return true
    end if
end procedure

Algorithm 6 : FLTP

Output: Proposal "createFLNFT" ***created***
procedure Generate_FLNFT(t)
    Create GM_t ▷ $t = 0$
    $GMCID \leftarrow$ **Store** GM_t on IPFS
    Create $FLNFT\_Metadata_t$ for GM_t
    $tokenURI \leftarrow$ **Store** $FLNFT\_Metadata_t$ on IPFS
    Call $MultiSigC.proposecreateFLNFT(GMCID, tokenURI)$
end procedure
Output: Proposal "Initiate_LMUs" ***created***
procedure Initiate_LMuploads($t+1$)
    Call $MultiSigC.propose$ $(selector, t+1)$ ▷ for "Initiate_LMUs"
end procedure
Output: Proposal "Cease_LMUs" ***created***
procedure Halt_LMuploads($t+1$)
    Call $MultiSigC.propose$ $(selector, t+1)$ ▷ for "Cease_LMUs"
end procedure
Output: Proposal "setLMUVDRF" ***created***
procedure Configure_LMUVDRF($t+1$)
    Call $MultiSigC.propose$ $(selector, t+1)$ ▷ for "setLMUVDRF"
end procedure
Output: Proposal "UpdateGM" ***created***
procedure Aggregate_LMUs($t+1$)
    Create GM_{t+1} using [9]
    $GMCID \leftarrow$ **Store** GM_{t+1} on $IPFS$
    Create $FLNFT\_Metadata_{t+1}$ for GM_{t+1}
    $tokenURI \leftarrow$ **Store** $FLNFT\_Metadata_{t+1}$ on IPFS
    Call $MultiSigC.proposeUpdateGM(t+1, GMCID, tokenURI)$
end procedure

If a *proposal* lacks sufficient approvals due to inaccuracies in $tokenURI$ or $GMCID$, FLTP can submit revised proposals. To discard inaccurate ones, FLTP submits a "closeProposal" transaction with the $proposalID$, closing it for future accurate submissions. This process is detailed in the procedure $closeProposal$ (Algo. 4).

F. Execution Workflow of DAO-FL framework

In this subsection, we explore the execution workflow of the DAO-FL framework for a complete GI t, as depicted in Fig. 5. The following is a outline of the sequential flow:

- Step 1: The Regulator deploys the FLNFTC for the FL ecosystem, providing three arguments: "Federated

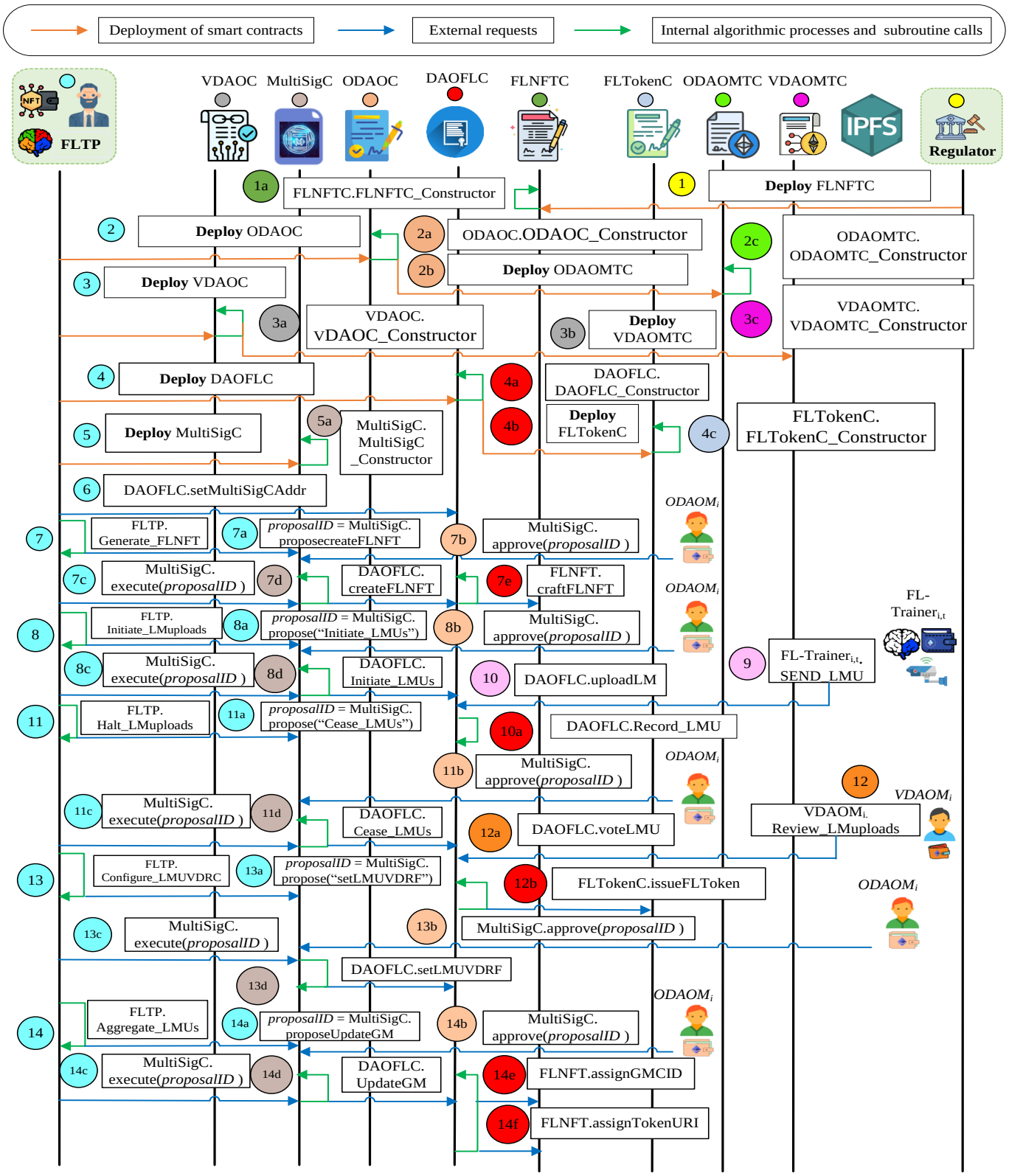

Fig. 5. DAO-FL: Simplified execution workflow.

Learning NFT" as the name, "FLNFT" as the symbol, and a base_URI for the $tokenURI$ of FL-NFTs. Ownership of FLNFTC is subsequently transferred to the $Regulator$. This process is summarized in the procedure $FLNFTC\_Constructor$ (Algo. 5).

- Step 2: FLTP deploys ODAOC, specifying two candidate ODAOMs ($ODAOM_i$) and a base_URI for $tokenURI$ of ODAOMTs. The procedure $ODAOC\_Constructor$ (Algo. 7) transfers ownership of ODAOC to FLTP. ODAOC then deploys ODAOMTC with specified parameters (name, symbol, base_URI) as outlined in procedure $ODAOMTC\_Constructor$ (Algo. 8), transfer-

Algorithm 7 : ODAOC **Owner & Deployer:** $FLTP$

Output: ODAOC ***deployed***, $(FLTP, member1, member2) \in ODAO$
procedure $ODAOC\_Constructor($ $member1$, $member2$, *base_URI*)
  Set $ODAOC.owner = FLTP.address$
  Deploy $ODAOMTC$ ("Orchestrator-DAOMT","ODAOMT", *base_URI*)
  Call $ODAOMTC.mint(recipent)$ **for** $recipent$ **in** $[FLTP, member1, member2]$
end procedure

Algorithm 8 : ODAOMTC **Owner & Deployer:** $ODAOC$

Output: ODAOMTC ***deployed***
procedure ODAOMTC_Constructor(*_name*, *_symbol*, *base_URI*)
  $ODAOMTC \leftarrow \{owner$: $ODAOC.address$, $name$: $\_name$, $symbol$: $\_symbol$, $base\_URI$: $base\_URI\}$
end procedure
Caller: ODAOC **Modifier:** onlyOwner() **Output:** [*ODAOMT **minted***]
procedure $mint$(**address** $recipent$)
  if $candidate \notin ODAO$ **then**
    Mint $ODAOMT$ for $recipent$
  end if
end procedure

Algorithm 9 : VDAOC **Owner & Deployer:** $FLTP$

Output: VDAOC ***deployed***, $(FLTP, member1, member2) \in VDAO$
procedure VDAOC_Constructor($member1$, $member2$, *base_URI*)
  Set $VDAOC.owner = FLTP.address$
  Deploy $VDAOMTC$ ("Validation-DAOMT","VDAOMT", *base_URI*)
  Call $VDAOMTC.mint(recipent)$ **for** $recipent$ **in** $[FLTP, member1, member2]$
end procedure

ring ODAOMTC's ownership to ODAOC. ODAOC mints ODAOMTs for FLTP and the two members using the procedure $mint$ (Algo. 8). ODAOMs can then manage membership enrollment and expulsion within ODAOC, as defined in Section IV-B and Section IV-C, respectively.

- Step 3: FLTP deploys VDAOC following the procedure $VDAOC\_Constructor$ (Algo. 9), adding two initial VDAO members ($VDAOM_i$) and setting a base URI for the $tokenURI$ of VDAOMTs. Ownership of VDAOC is transferred to FLTP. Through the procedures $VDAOMTC\_Constructor$ and $mint$ (Algo. 10), VDAOC deploys VDAOMTC using specified parameters (name, symbol, base_URI) and transfers ownership to VDAOC. VDAOMTs are minted for FLTP and the two members. Once VDAOC is deployed, VDAOMs manage membership enrollment and expulsion operations within VDAOC.
- Step 4: FLTP deploys DAOFLC, passing the addresses of FLNFTC, ODAOC, and VDAOC as arguments, and transfers ownership of DAOFLC. DAOFLC then deploys FLTokenC with a designated name and symbol for FL-Tokens, transferring FLTokenC's ownership to DAOFLC. This process is detailed in the procedures $DAOFLC\_Constructor$ (Algo. 11) and $FLTokenC\_Constructor$ (Algo. 15).
- Step 5: FLTP deploys the MultiSigC and transfers its ownership, as outlined in procedure $MultiSigC\_Constructor$ (Algo. 12).
- Step 6: FLTP submits the "setMultiSigCAddr" transaction to DAOFLC with MultiSigC's address. The procedure $setMultiSigCAddr$ (Algo. 11) outlines this step. Afterward, MultiSigC can execute transactions within DAOFLC.

Algorithm 10 : VDAOMTC **Owner & Deployer:** $VDAOC$

Output: VDAOMTC ***deployed***
procedure VDAOMTC_Constructor(*_name*, *_symbol*, *base_URI*)
  $VDAOMTC \leftarrow \{owner$: $ODAOC.address$, $name$: $\_name$, $symbol$: $\_symbol$, $base\_URI$: $base\_URI\}$
end procedure
Caller: VDAOC **Modifier:** onlyOwner() **Output:** [*VDAOMT **minted***]
procedure $mint$(**address** $recipent$)
  if $candidate \notin VDAO$ **then**
    Mint $VDAOMT$ for $recipent$
  end if
end procedure

- Step 7: Following procedure $Generate\_FLNFT$ (Algo. 6), FLTP constructs the "preliminary GM parameters" for the FLT, stores them on IPFS, and generates a CID called $GMCID$, representing GM_t for $t = 0$. FLTP also uploads additional files (instructions for FLT, LMUs, reward criteria, etc.) to IPFS, forming a JSON-encoded metadata known as $FLNFT\_Metadata_t$. This metadata is uploaded to IPFS, resulting in a URI called $tokenURI$. FLTP then initiates procedure $proposecreateFLNFT$ ($propose$) in Algo. 4, starting the multi-signature process (Section IV-E) for the "createFLNFT" $proposal$. Upon its execution, DAOFLC mints the FL-NFT on FLNFTC for FLTP, as shown in procedures $createFLNFT$ (Algo. 11) and $craftFLNFT$ (Algo. 5), setting the FL-NFT's properties, including the $OrchestratorAddress$.
- Step 8: The FLTP triggers procedure $Initiate\_LMuploads$ (Algo. 6) to commence the LMUs on the DAOFLC. It then initiates procedure $propose$ (Algo. 4) with parameters such as $selector$ and GI $t+1$, where $selector$ is derived from the "Initiate_LMUs" function signature [28] in the DAOFLC. This starts the multi-signature process for the "Initiate_LMUs" $proposal$ (Section IV-E). During its execution, procedure $Initiate\_LMUs$ (Algo. 11) verifies that status of the $DAOFLC.LMUactiveF$ flag is false. If true, it indicates LMUs are accepted; if false, it updates it to true and emits the $LMUsInitiated(t+1)$ event, signaling the initiation of LMUs for GI $t+1$. FL-Trainers monitor this event to submit their LMUs.
- Step 9: $FLTrainers_{t+1}$ concurrently initiate procedure $SEND\_LMU$ (Algo. 13) to commence their LMUs on DAOFLC. Each $FLTrainer_{i,t+1}$ retrieves the latest GM CID from $DAOFLC.GMCID$ and downloads the corresponding GM (GM_t) from IPFS. Utilizing their local private dataset $D_{i,t+1}$, $FLTrainer_{i,t+1}$ computes their local model $LMU_{i,t+1}$ as [10], [17]:

$$\boldsymbol{w}_{t+1}^{i} \leftarrow \boldsymbol{w}_t - \eta g_i, \quad \forall i. \tag{6}$$

Where g_i is the local gradient of $FLTrainer_{i,t+1}$ on $D_{i,t+1}$, $\boldsymbol{w}_t$ is the global parameter, η is learning rate, and $\boldsymbol{w}_{t+1}^{i}$ is the local parameter. Subsequently, $LMU_{i,t+1}$ is stored on IPFS, generating the associated CID $LMCID$. Additionally, JSON-encoded meta-data for $LMU_{i,t+1}$ is created and stored on IPFS, resulting in the URI $LMURI$. $FLTrainer_{i,t+1}$ submits its LMU to DAOFLC using procedure $uploadLM$ (Algo. 11), with

Algorithm 11 : DAOFLC **Owner & Deployer:** $FLTP$

Output: DAOFLC, FLTokenC ***deployed***
procedure DAOFLC_Constructor($FLNFTC.address$, $ODAOC.address$, $VDAOC.address$)
  Set $DAOFLC.owner = FLTP.address$
  Deploy $FLTokenC$ ("Federated Learning Token", "FLToken")
end procedure

Caller: $FLTP$ **Modifier:** onlyOwner()
Output: $DAOFLC.MultiSigCAddr$ ***set***
procedure setMultiSigCAddr(MultiSigC.address)
  Set $DAOFLC.MultiSigCAddr = MultiSigC.address$
end procedure
Caller: $MultiSigC$ **Modifier:** onlyMultiSigC() **Output:** FLNFT ***minted***
procedure createFLNFT($tokenURI$, $GMCID$)
  $FLNFTID$ = **call** $FLNFTC.craftFLNFT$ ($tokenURI$, $GMCID$)
  $DAOFLC \leftarrow \{FLNFTID$: $FLNFTID$, $GMCID$: $GMCID\}$
end procedure
Caller: $MultiSigC$ **Modifier:** onlyMultiSigC() **Output:** LMUs ***initiated***
procedure Initiate_LMUs($t+1$)
  if $DAOFLC.LMUactiveF == false$ **then**
    Set $DAOFLC.LMUactiveF = true$
    Emit $DAOFLC.LMUsInitiated(t+1)$
  end if
end procedure
Caller: $MultiSigC$ **Modifier:** onlyMultiSigC() **Output:** LMUs ***ceased***
procedure Cease_LMUs(t+1)
  if $LMUactiveF == true$ **then**
    Set $LMUactiveF = false$; **Set** $LMUC[t+1] = true$
    Emit $LMUsCeased(t+1)$
  end if
end procedure
Caller: $MultiSigC$ **Modifier:** onlyMultiSigC()
Output: $DAOFL.LMUVDRF[t+1]$ ***set***
procedure setLMUVDRF($t+1$)
  Set $LMUVDRF[t+1] = true$
end procedure
Caller: $MultiSigC$ **Modifier:** onlyMultiSigC() **Output:** FLNFT ***updated***
procedure UpdateGM($t+1$, $GMCID$, $tokenURI$)
  GMCIDsuccessF = **Call** $FLNFTC.assignGMCID($ $GMCID, FLNFTID$)
  TokenURIsuccessF = **Call** $FLNFTC.assignTokenURI($ $tokenURI, FLNFTID$)
  if GMCIDsuccessF **and** TokenURIsuccessF **then**
    Emit $GMupdated(t+1, GMCID, tokenURI)$
    $DAOFLC \leftarrow \{tokenURI$: $tokenURI$, $GMCID$: $GMCID\}$
    Set $GIC[t+1] = true$
  end if
end procedure
Caller: $FLTrainer_{i,t+1}$ **Output:** LMU ***uploaded*** and ***recorded***
procedure uploadLM($LMCID$, $LMURI$, $t+1$)
  if DAOFLC.Authenticate_LMU($LMCID$, $LMURI$, $t+1$, $FLTrainer_{i,t+1}.address$) **then**
    Call $DAOFLC.Record\_LMU(LMCID,$ $LMURI, t+1, FLTrainer_{i,t+1}.address$)
  end if
end procedure
Output: LMU ***recorded*** for $FLTrainer_{i,t+1}$
procedure Record_LMU($LMCID$, $LMURI$, $t+1$, $FLTrainer_{i,t+1}$)
  LM = **Create** *new* $LMUs[t+1][FLTrainer_{i,t+1}.address]$
  $LM \leftarrow \{status$: "Submitted", $LMCID$: $LMCID$, $LMURI$: $LMURI$, $approvalvotes$: 0, $denialvotes$: 0, $voters$: $\emptyset\}$
end procedure
Caller: $VDAOM_i$ **Modifier:** onlyVDAOM()
Output: LM ***updated***, [LM ***rewarded***]
procedure voteLMU($FLTrainer_{i,t+1}.address$, $t+1$, V_i)
  Require:$VDAOM_i \notin$ $LMUs[t+1][FLTrainer_{i,t+1}.address].voters$
  $LM \leftarrow LMUs[t+1][FLTrainer_{i,t+1}.address]$
  if $V_i == true$ **then**
    Add Approval vote for $VDAOM_i$ in LM
  else
    Add Deny vote for $VDAOM_i$ in LM
  end if
  Count $LM.approvalvotes$ and $LM.denialvotes$
  if $LM.approvalvotes > (Q = 60\% * n(VDAOMT))$ **then**
    Call $FLTokenC.issueFLToken(FLTrainer_{i,t+1})$
    Set $LM.status$="Rewarded"
  else if $LM.denialvotes > Q$ **then**
    Set $LM.status$="Denied"
  end if
end procedure

Algorithm 12 : MultiSigC **Owner & Deployer:** $FLTP$

Output: MultiSigC ***deployed***
procedure MultiSigC_Constructor($DAOFLC.address$, $ODAOC.address$)
  Set $MultiSigC.owner = FLTP.address$
end procedure

$LMCID$ and $LMURI$. DAOFLC may impose a limit on the number of LMUs permitted for GI $t+1$.

- Step 10: The procedure $uploadLM$ (Algo. 11) is initiated by $FLTrainer_{i,t+1}$. The DAOFLC.Authenticate_LMU function validates $LMU_{i,t+1}$ and may reject it if the LMUs limit is exceeded. If valid, $LMU_{i,t+1}$ is added to the LMUs for GI $t+1$ and associated with $FLTrainer_{i,t+1}$ through procedure $FLTPC.Record\_LMU$ (Algo. 11). The LM properties, including approval and denial votes, are initialized to 0, the voter list is empty, and the LM's status is marked as "Submitted."
- Step 11: The FLTP initiates the procedure $Halt\_LMuploads$ (Algo. 6) to stop LMUs on the DAOFLC. This triggers the procedure $propose$ (Algo. 4) with arguments like $selector$ and $t+1$, where $selector$ is linked with the DAOFLC's "Cease_LMUs" function. This starts the multi-signature process (Section IV-E) for the "Cease_LMUs" $proposal$. The proposal's execution activates the $Cease\_LMUs$ procedure (Algo. 11). If $LMUactiveF$ is true, it is set to false. The $LMUceased(t+1)$ event is emitted, the $LMUC$ flag is marked as true, and LMUs for GI $t+1$ are halted.
- Step 12: After LMUs are ceased for $t+1$, VDAOMs in VDAO concurrently initiate the procedure $Review\_LMuploads$ (Algo. 14). In this procedure, each $VDAOM_i$ downloads the LM uploaders' addresses using the function DAOFLC.Fetch_LMUx($t+1$). For each $FLTrainer_{i,t+1}$ in the fetched list, the VDAOM downloads the corresponding LMU ($LMU_{i,t+1}$) using the function DAOFLC.Fetch_LMU($t+1$, $FLTrainer_{i,t+1}$). The $VDAOM_i$ checks $LMU_{i,t+1}$ and casts an approval or denial vote by invoking procedure $DAOFLC.voteLMU$ (Algo. 11) with a boolean vote argument V_i. True signifies approval, while false indicates disapproval for $LMU_{i,t+1}$. The total approval and denial votes are counted as

$$LM_{approvalvotes} = \sum_{V_i \in LM.voters} \mathbf{1}_{V_i == \text{true}}, \quad (7)$$

and

$$LM_{denialvotes} = \sum_{V_i \in LM.voters} \mathbf{1}_{V_i == \text{false}} \quad (8)$$

respectively. The quorum Q is determined. If the $LM_{approvalvotes}$ exceed the Q, the procedure $FLTokenC.issueFLToken$ (Algo. 15) is utilized to issue an FL-Token for $FLTrainer_{i,t+1}$, LM's status is set to "Rewarded". However, if the $LM_{denialvotes}$ exceed the Q, the LM's status is set to "Denied".
- Step 13: The FLTP initiates the procedure $Configure\_LMUVDRC$ (Algo. 6) using the $selector$ of the "setLMUVDRF" function in the DAOFLC and $t+1$. This triggers the multi-signature process (Section

Algorithm 13 : FL-Trainer $FLTrainer_{i,t+1}$

Output: LMU ***generated*** and ***uploaded*** by $FLTrainer_{i,t+1}$
procedure SEND_LMU
  Get $DAOFLC.GMCID$
  Download $GM_t \leftarrow$ IPFS using $DAOFLC.GMCID$
  Generate $LMU_{i,t+1}$ using [6]; $LMCID$ = **Store** $LMU_{i,t+1}$ on IPFS
  Create $LMURI$ for $LMU_{i,t+1}$; $LMURI$ = **Store** $LMURI$ on IPFS
  Call $DAOFLC.uploadLM(LMCID, LMURI, t+1)$
end procedure

Algorithm 14 : VDAO member $VDAOM_i$

Output: LMUs ***voted*** by $VDAOM_i$
procedure Review_LMuploads
  foreach $FLTrainer_{i,t+1}$ in DAOFLC.Fetch_LMUx($t+1$)
    $LMU_{i,t+1}$ = **Call** DAOFLC.Fetch_LMU($t+1$, $FLTrainer_{i,t+1}.address$)
    Call $DAOFLC.voteLMU(FLTrainer_{i,t+1}.address, t+1, V_i)$
  end foreach
end procedure

Algorithm 15 : FLTokenC **Owner & Deployer:** $DAOFLC$

Output: FLTokenC ***deployed***
procedure FLTokenC_Constructor(*_name*, *_symbol*)
  $FLTokenC \leftarrow \{owner: DAOFLC.address, name: \_name, symbol: \_symbol\}$
end procedure
Output: FLToken ***minted*** for $FLTrainer_{i,t+1}$
procedure issueFLToken($FLTrainer_{i,t+1}.address$)
  Mint $1 * 10^{18}$ $FLToken$ for $FLTrainer_{i,t+1}$
end procedure

IV-E) for the "setLMUVDRF" *proposal*, by invoking the procedure *propose* (Algo. 4). During proposal execution, the procedure $setLMUVDRF$ (Algo. 11) is called to set the $LMUVDRF(t+1)$ flag, indicating the completion of the LMUs' verification, denial, or reward process for GI $t+1$.

- Step 14: The FLTP initiates the $Aggregate\_LMUs$ procedure (Algo. 6). The approved and rewarded LMUs from previous steps are denoted as $\hat{LMU}_{t+1}$. The FLTP computes GM_{t+1} using federated averaging (FedAvg) as [10], [17]:

$$\boldsymbol{w}_{t+1} \leftarrow \sum_{i \in \hat{LMU}_{t+1}} \frac{n_i}{n} \boldsymbol{w}^i_{t+1} \tag{9}$$

where $\boldsymbol{w}^i_{t+1}$ is local parameter, $\boldsymbol{w}_{t+1}$ is global parameter, $n_i = |\mathcal{D}_i|$, and $n = \left|\bigcup \mathcal{D}_i\right|$. GM_{t+1} is stored on IPFS, generating a CID referred to as $GMCID$. The updated meta-data, encoded in JSON format as $FLNFT\_Metadata_{t+1}$, is also stored on IPFS, yielding a URI called $tokenURI$. The FLTP initiates the procedure $proposeUpdateGM$ (*propose*) (Algo. 4) with arguments $t+1$, $GMCID$, and $tokenURI$, This starts the multi-signature process outlined in Section IV-E for the "UpdateGM" *proposal*. During this process, ODAOMs aggregate $\hat{LMU}_{t+1}$ according to predefined guidelines and approve the proposal if it is valid, certifying its authenticity. When "UpdateGM" *proposal* is executed, the procedure $UpdateGM$ (Algo. 11) is called, setting the $GMCID$ and $tokenURI$ of the FL-NFT by invoking procedures $FLNFTC.assignGMCID$ and $FLNFTC.assignTokenURI$ (Algo. 5) [17]. These procedures also verify and ensure the uniqueness of $GMCID$ and $tokenURI$ for all FL-NFTs. The DAOFLC emits the event $DAOFLC.GMupdated$, and the $GIC[t+1]$ flag indicates the completion of GI $t+1$.

Algorithm 16 : FL-NFT's transfer

Caller: $FLTP$ **Output:** [Contracts Ownership ***transferred***]
procedure FLNFT_TRANSFER(new_owner)
  Require: new_owner != $FLTP.address$
  Call FLNFTC.transferFrom($FLTP.address$, new_owner, $FLNFTID$)
  $contracts$ = [$ODAOC$, $VDAOC$, $MultiSigC$, $DAOFLC$]
  for all $contract$ in $contracts$ **do**
    Call $contract.transferOwnership(new\_owner)$
  end for
end procedure

Step 1 of the above execution workflow is carried out once by the Regulator to establish the FL marketplace ecosystem. For each FLT, Steps 2-7 are repeated to prepare the FL decentralized orchestrating space using the DAO-FL framework. Additionally, Steps 8-14 are reiterated for each GI $t+1$ within an FLT.

G. Commercializing GM and Transferring ownership

The GM is tokenized to streamline FL processes and facilitate commercialization on platforms such as OpenSea. Trading involves transferring the FL-NFT of GM to the buyer. In DAO-FL, the FLTP, which holds the FL-NFT, also controls contracts including DAOFLC, MultiSigC, ODAOC, and VDAOC. To transfer GM's ownership to a new owner, the FLTP initiates the procedure $FLNFT\_Transfer$ (Algo. 16). This process encompasses transferring the FL-NFT as well as the ownership of DAOFLC, MultiSigC, ODAOC, and VDAOC to the new proprietor.

V. IMPLEMENTATION, DEPLOYMENT, AND EVALUATION

This section presents the implementation, deployment, and evaluation of the DAO-FL framework.

A. Implementation and Deployment

The smart contracts are developed using Solidity [29], with Surya [30] used to visualize their inheritance hierarchy. To support membership in ODAO and VDAO, we initially considered the NTT token standard (EIP-4671 [31]), but due to its early-stage limitations, we have developed a custom smart contract "DAOMTC" to implement DAOMTs. The inheritance graph in Appendix D shows that DAOMTC inherits from OpenZeppelin's "Ownable" and "ERC165" contracts and implements the IERC721Metadata interface. Since DAOMTs are NTTs, certain functions of the IERC721 interface are not applicable but are retained for compatibility with NFT platforms like OpenSea. To streamline membership management in both ODAO and VDAO, we have developed a smart contract named DAOC. By providing generalized procedures for adding or removing members, DAOC is designed to serve the needs of both DAOs. The DAOC contract extends a customized version of the "Ownable" contract [32], which itself inherits from the "Context" contract [32]. The class diagram for DAOMTC and DAOC is provided in Appendix B.

ODAO and VDAO are distinct DAOs implemented in ODAOC and VDAOC, using ODAOMTs and VDAOMTs

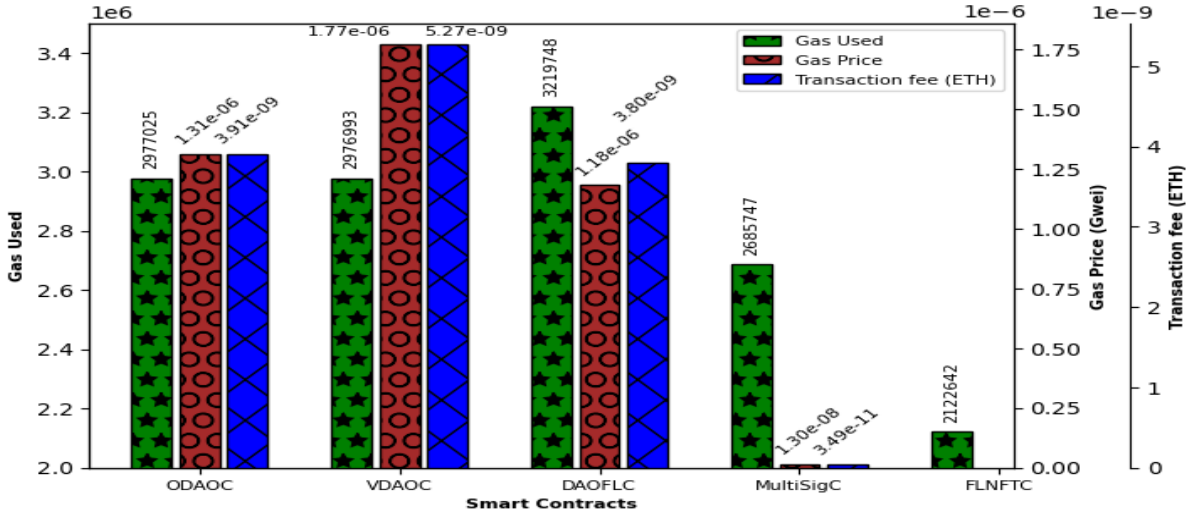

Fig. 6. Gas Used, Gas Price, and transaction fee (in ETH) for the deployment of smart contracts.

TABLE III
PARAMETERS

Parameter	Value on Sepolia
$Regulator.etherscan$	https://sepolia.etherscan.io/address/0x8fa37ecf3d89361e60e7e6adf55485ae62cd72b2
$FLTP.etherscan$	https://sepolia.etherscan.io/address/0xa0969AeA747c336b49256CFC4Cc2F6E265F6B722
$FLNFTC.etherscan$	https://sepolia.etherscan.io/address/0x37d18bd11e20774e9BE7c22647156564975CAe6b
$ODAOC.etherscan$	https://sepolia.etherscan.io/address/0xf002f304Cb1C34b40d59347472f2f68Fc882e61f
$ODAOMTC.etherscan$	https://sepolia.etherscan.io/address/0xDfF3E610ce7DCb727150E1351c44e58154E28108
$VDAOC.etherscan$	https://sepolia.etherscan.io/address/0x1d9Cebd90Aa66068cD9FD3d75479DbDeDA65ebeB
$VDAOMTC.etherscan$	https://sepolia.etherscan.io/address/0x5303b5a16655C69D7914cf6fcdF5A5429C41279F
$DAOFLC.etherscan$	https://sepolia.etherscan.io/address/0x21314B8830c7FE06d0B0DAe0c7935794D77FD429
$FLTokenC.etherscan$	https://sepolia.etherscan.io/address/0x13C3A1a153F7C50a018177aeaC5D70D98A3B2c2C
$MultiSigC.etherscan$	https://sepolia.etherscan.io/address/0x7001b7f257EEDF4b970577c63095909916BD0cc0
$FLTrainer_{1,1}.etherscan$	https://sepolia.etherscan.io/address/0xff0e2447422da30927fd079d75dd985cf0cd21e1

as membership tokens, respectively. These tokens are implemented in ODAOMTC and VDAOMTC. As shown in Appendix D, ODAOC and VDAOC inherit from DAOC, while ODAOMTC and VDAOMTC inherit from DAOMTC. UML diagrams for ODAOC, ODAOMTC, VDAOC, and VDAOMTC are detailed in Appendix C.

FLNFTC inherits from the ERC721Enumerable standard [33] and the "Ownable" contract [32]. Appendix D depicts the inheritance graph of FLNFTC. FLTokenC derives from "Ownable" contract and the OpenZeppelin ERC-20 implementation [34], [35]. DAOFLC and MultiSigC also inherit from the "Ownable" contract. Appendix E illustrates the inheritance graph for DAOFLC, MultiSigC, and FLTokenC. UML diagrams for DAOFLC, FLTokenC, FLNFTC, and MultiSigC are in Appendix C.

The smart contracts were compiled using Hardhat [36] and deployed on the Sepolia testnet [37]. For transparency, contract verification was conducted using the ETHERSCAN_API_KEY [38]. Gas usage, gas price, and transaction fees are illustrated in Fig. 6. The gas consumed by ODAOMTC, VDAOMTC, and FLTokenC is already accounted for within the consumption of ODOAC, VDOAC, and DAOFLC, respectively. Additionally, due to network congestion, FLNFTC incurred a higher gas price of 0.15 Gwei, resulting in a transaction fee of 0.00032 ETH, which is not depicted in Fig. 6.

Table III provides Etherscan links for key entities (Regulator, FLTP, $FLTrainer_{1,1}$) and smart contracts deployed on the Sepolia network. Users can explore event logs, transaction logs, and verified contract codes via Etherscan [17]. Our paper focuses on establishing a decentralized ecosystem for IV and OV of the FL process through multi-signature wallets and DAOs. We used the MNIST, Fashion-MNIST, and UNB ISCX VPN-NonVPN network traffic [39] datasets for training local and global models. Due to space constraints, we have omitted details on model configuration, accuracy, and data allocation. Some repetitive quorum-related transactions are also omitted in subsequent figures for brevity.

Since the procedures for member enrollment and expulsion are identical for ODAOC and VDAOC, we present the implementation results for ODAOC. Fig. 7 shows the transaction list for a "Join Proposal" (JP), which includes the "proposeJoin" transaction initiated by $ODAOM_p$ and the "voteJoin" transactions from other ODAOMs. Fig. 8 illustrates the minting of ODAOMT upon reaching quorum, alongside the "JPapproved" event indicating JP approval and the "Transfer" event indicating ODAOMT transfer to the $candidate$, emitted by ODAOMTC. Similarly, Fig. 9 presents the transaction list for a "Kick Proposal" (KP), including the "proposeKick" transaction and "voteKick" transactions. Fig. 10 demonstrates the burning of ODAOMT upon quorum, with events indicating KP approval and the burning of ODAOMT owned by the

Transaction Hash	Method	Block	Age	From		To
0x08d720a710...	Vote Join	3815822	467 days ago	0x7e727f7E...4bd5e7676	IN	0xf002f304...Fc882e61f
0x18f0159d577...	Vote Join	3815821	467 days ago	0xe319A0Fd...D736909e5	IN	0xf002f304...Fc882e61f
0x93c3814116f...	Propose Join	3815817	467 days ago	0xf0A229BD...B40F56194	IN	0xf002f304...Fc882e61f

Fig. 7. Transaction sequence ($DAOC.proposeJoin$ and $DAOC.voteJoin$) for a "Join Proposal" on ODAOC (https://sepolia.etherscan.io/address/0xf002f304Cb1C34b40d59347472f2f68Fc882e61f), [Block 3815817-3815822].

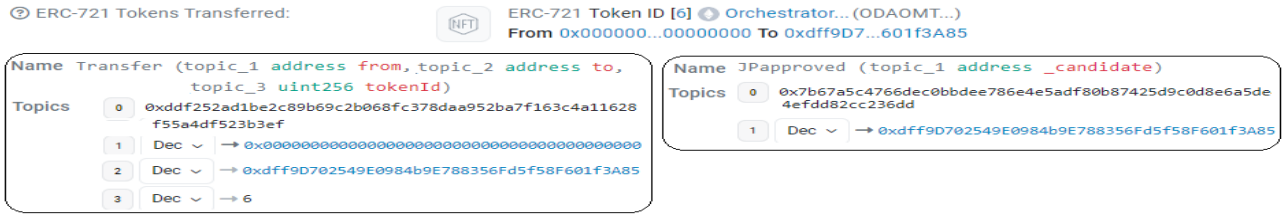

Fig. 8. Minting of ODAOMT after reaching the quorum of approval votes for "Join Proposal" and corresponding events emitted on ODAOC (https://sepolia.etherscan.io/tx/0x08d720a7101486f789952ce09e72cb0bf56ce8863994d3eacf957a29d0a1ea6a).

Transaction Hash	Method	Block	Age	From		To
0x7de873fc9bd...	Vote Kick	3815828	467 days ago	0xdff9D702...F601f3A85	IN	0xf002f304...Fc882e61f
0xb4de764d43...	Vote Kick	3815827	467 days ago	0x7e727f7E...4bd5e7676	IN	0xf002f304...Fc882e61f
0x0d2aa97fc65...	Vote Kick	3815824	467 days ago	0xa0969AeA...265F6B722	IN	0xf002f304...Fc882e61f
0xc6cd5ba3fba...	Propose Kick	3815823	467 days ago	0xf0A229BD...B40F56194	IN	0xf002f304...Fc882e61f

Fig. 9. Transaction sequence ($DAOC.proposeKick$ and $DAOC.voteKick$) for a "Kick Proposal" on ODAOC (https://sepolia.etherscan.io/address/0xf002f304Cb1C34b40d59347472f2f68Fc882e61f), [Block 3815823-3815828].

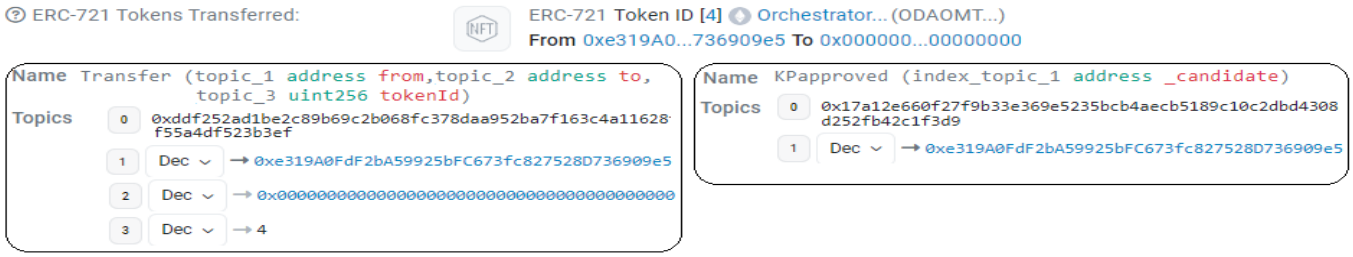

Fig. 10. Burning of ODAOMT after reaching the quorum of approval votes for "Kick Proposal" and corresponding events emitted on ODAOC (https://sepolia.etherscan.io/tx/0x7de873fc9bdfb1fca45ad560430eff5ee4778e821fd1e8d981c12a6f1c099da3).

Transaction Hash	Method	Block	Age	From		To
0x8fd223394c0b4e597...	Execute	3829547	1 min ago	0xa0969A...65F6B722	IN	0x7001b7...16BD0cc0
0xe8e49c06d64ee2c9...	Approve	3829546	1 min ago	0xdff9D7...601f3A85	IN	0x7001b7...16BD0cc0
0xe7ae6e70d8fe92330...	Proposecreat...	3829542	2 mins ago	0xa0969A...65F6B722	IN	0x7001b7...16BD0cc0

Fig. 11. Transaction sequence for the creation and execution of the "createFLNFT" proposal on MultiSigC, along with emitted events (https://sepolia.etherscan.io/address/0x7001b7f257EEDF4b970577c63095909916BD0cc0), Block [3829542-3829547].

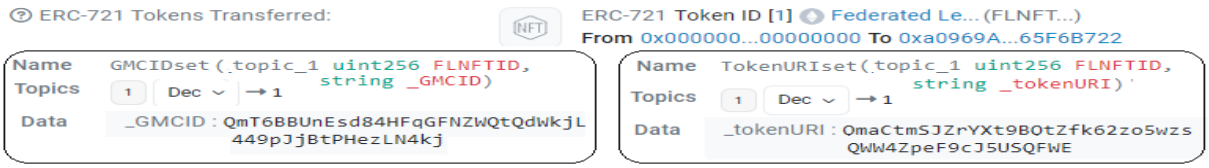

Fig. 12. Minting of FL-NFT and emitted events during the execution of the "createFLNFT" proposal (https://sepolia.etherscan.io/tx/0x93e76ce42d9b76f6b4ede511e262e7ac9d77e5079f2cd0171e8e2e554d231a7a).

candidate.

Continuing in this subsection, we present the implementation of the DAO-FL framework, following the steps in Section IV-F. Fig. 11 illustrates the creation of a "createFLNFT" *proposal* by FLTP using the procedure $FLTP.Generate\_FLNFT$ through the "proposecre-

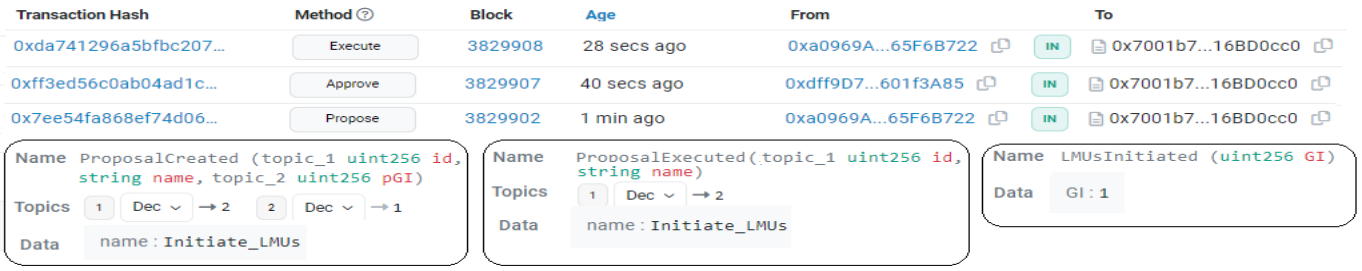

Fig. 13. Execution of the "Initiate_LMUs" proposal by FLTP, and emitted events by MultiSigC and DAOFLC (https://sepolia.etherscan.io/address/0x7001b7f257EEDF4b970577c63095909916BD0cc0), Block [3829902-3829908].

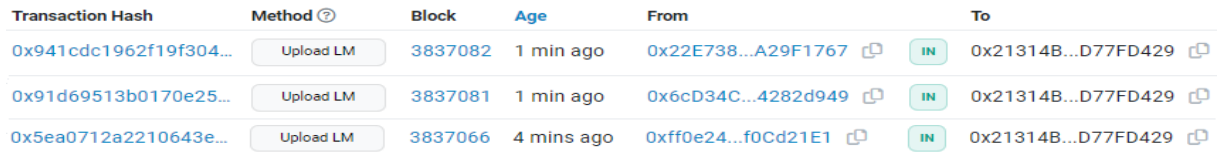

Fig. 14. Uploading of LM on DAOFLC by $FLTrainers_{t+1}$ (https://sepolia.etherscan.io/address/0x21314B8830c7FE06d0B0DAe0c7935794D77FD429) , Block [3837066-3837082].

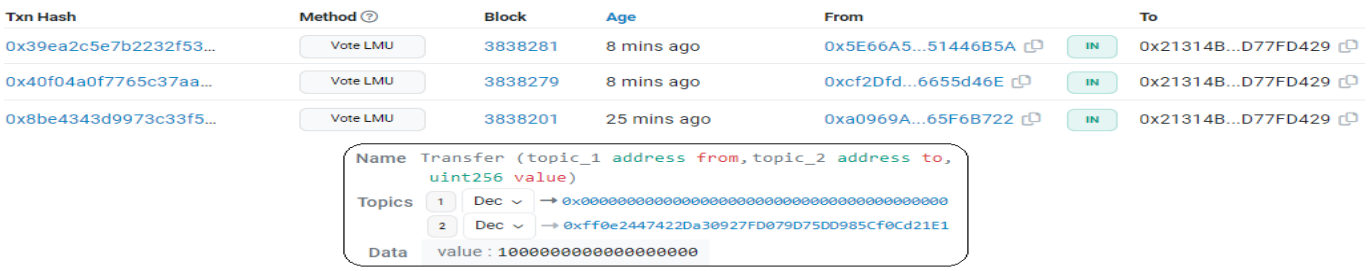

Fig. 15. Decentralized input verification of LMUs by VDAOMs for the FL process, minting of FL-Token and other events emitted (https://sepolia.etherscan.io/address/0x21314B8830c7FE06d0B0DAe0c7935794D77FD429), Block [3838201-3838281].

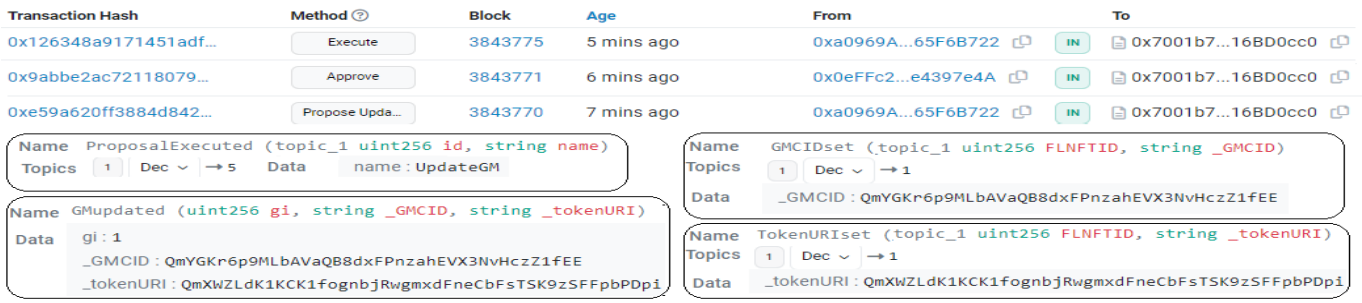

Fig. 16. Creation and execution of proposal "UpdateGM" after DOV by ODAOMs (https://sepolia.etherscan.io/address/0x7001b7f257EEDF4b970577c63095909916BD0cc0) and events emitted, Block [3843770-3843775].

ateFLNFT" transaction on MultiSigC. It includes one of the "approve" transactions by ODAOMs and the execution of the "createFLNFT" *proposal* by FLTP upon reaching quorum, with the corresponding events emitted by MultiSigC shown. Fig. 12 demonstrates the minting of FL-NFT after executing the "createFLNFT" *proposal*, displaying the emitted events from FLNFTC, including GMCIDset and TokenURIset. Fig. 13 shows the creation and execution of the "Initiate_LMUs" *proposal* by FLTP after ODAOMs' approval. This figure also includes emitted events such as ProposalCreated and ProposalExecuted by MultiSigC, along with LMUsInitiated by DAOFLC. Following the LMUsInitiated event, $FLTrainers_{t+1}$ upload LMs via the "uploadLM" transaction on DAOFLC, as depicted in Fig. 14.

The creation and execution of the "Cease_LMUs" *proposal* will be omitted. After its execution, VDAOMs perform IV for the FL process by initiating "voteLMU" transactions, as shown in Fig. 15. The Successful validation of an LMU, results in minting FL-Tokens, as indicated by the "Transfer" event emitted by FLTokenC for a $FLTrainer_{i,t+1}$. We will omit the illustration of the execution of "setLMUADRF" *proposal*. However, after its execution, FLTP submits the "UpdateGM" *proposal* to MultiSigC. This *proposal* undergoes approval by ODAOMs as part of DOV and is ultimately executed as shown in Fig 16. The emitted events include ProposalExecuted by MultiSigC, GMupdated by DOAFLC, and GMCIDset and TokenURIset by FLNFTC, indicating the FL-NFT has been updated.

B. Evaluation on Threat Models

Vulnerabilities in information flows can occur at both input and output stages. Input vulnerabilities arise when submitted inputs deviate from prescribed policies. In the FL process, this can manifest as inaccurate or malicious LMs being accepted by

TABLE IV
AVERAGE TRANSACTION COST FOR CENTRALIZED VS DECENTRALIZED IV AND OV ON PUBLIC AND PRIVATE BLOCKCHAIN

		Input Verification	Output Verification
FL-Incentivizer (Centralized)	Public blockchain	0.000140322 ETH	0.00017406 ETH
	Private blockchain	0	0
DAO-FL (Decentralized)	Public blockchain	0	0
	Private blockchain	0.001080806 ETH	0.00159984 ETH

the compromised FL server for inclusion in the GM, jeopardizing its accuracy. Output vulnerabilities involve non-compliance of the produced outputs with information flow policies or post-production tampering. In the FL process, this includes aggregation attacks, where LMs are incorrectly aggregated into the GM. It also involves post-production tampering, where a malicious entity replaces the legitimate GM with a fraudulent version, further jeopardizing the integrity of the process.

Fig. 17 displays the test accuracy trends of DAO-FL and centralized-FL against input, output, and input & output attacks. The experiments utilize the MNIST and Fashion-MNIST datasets for image classification, along with the UNB ISCX VPN-NonVPN dataset [39] for network traffic classification [40], with parameters set to $E = 10$ local epochs and $N = 10$ FL-Trainers per GI. Fig. 17(a, c, d, f, g, i) highlight DAO-FL's robustness against input attacks, rejecting malicious LMs through DIV via VDAO, thus maintaining GM accuracy. DAO-FL closely matches the accuracy of attack-free-FL, especially near convergence. The slight accuracy drop in DAO-FL during input attacks compared to attack-free-FL is due to the latter's diverse accurate LMs, while DAO-FL's global parameters are biased towards approved LMs. In contrast, centralized-FL, relying on a single manipulable server, loses accuracy under input attacks. 17(b, c, e, f, h, i) show that DAO-FL maintains accuracy during output attacks due to ODAO's vigilance through DOV, which rejects malicious "UpdateGM" proposals. The ODAO enforces the FLTP for alternative accurate "UpdateGM" proposals. Centralized-FL suffers from accuracy deterioration due to tampering and aggregation attacks. These results illustrate DAO-FL's effectiveness against both input and output attacks.

Both input and output attacks in centralized FL result in decreased accuracy. After an attack, GM's original accuracy may not be recoverable. These attacks compromise accuracy, introduce bias, or disrupt the FL process, leading to learning failures like vanishing or exploding gradients. Such failures are evident for centralized-FL in Fig. 17(c,f) at epoch=10 and Fig. 17(i) at epoch=250 onwards. Preventing these attacks is crucial for the FL process's success.

C. Qualitative Evaluation and Discussion

DAO-FL ensures secure and democratic FL process management through decentralized governance involving regulators, FLTP, ODAO, and VDAO. DAO-FL promotes a decentralized approach to manage DAO membership for enrollment and expulsion through the use of smart contracts: ODAOC for ODAO and VDAOC for VDAO. It leverages the minting and burning of membership tokens, ensuring transparency and security. These operations rely on decentralized voting mechanisms, where stakeholders approve "Join Proposals" and "Kick Proposals" to control membership decisions. This ensures fairness and inclusivity in the membership process, although it may introduce delays due to the need for consensus.

DAO-FL is compatible with any FL algorithm, with LM and GM validation managed through DIV and DOV based on the algorithm's security protocol. DAO-FL enhances the trustworthiness of the FL process by allowing participants to verify LMUs quality and integrity via VDAOMs. As ODAOMTC and VDAOMTC supplies increase, decentralization in OV and IV of FL processes strengthens, respectively. In FL setups demanding high security and decentralization, the trade-off of longer times and higher transaction fees to reach quorum becomes acceptable as decentralization increases.

In DAO-FL, the ODAO approves proposals in a decentralized manner, while the execution remains the responsibility of the FLTP, leading to a partially decentralized orchestration of the FL process. To achieve full decentralization, substituting FLTP with an Executer-DAO and implementing a compatible multi-signature contract can enable the decentralized execution of approved proposals, resulting in a fully decentralized orchestration paradigm.

FL inherently protects raw data access, with the GM generally secure against sophisticated data leaks. However, LMs are still susceptible to inference attacks. In DAO-FL, VDAOMT authentication restricts LM access to authorized VDAO members via smart contract interfaces. Although the DAO-FL framework operates on a public blockchain, which poses risks from malicious actors, incorporating privacy-preserving techniques like differential privacy at the LM level can enhance data security and reduce the likelihood of data leaks.

The innovative principles and technologies in DAO-FL create a versatile framework that extends beyond its original application. DAOMTs can serve as universal proof of membership in various DAOs, while the proposed decentralized enrollment and expulsion schemes are relevant across multiple DAO implementations. The adaptability of smart contracts like MultiSigC and DAOFLC is significant; with careful adjustments to requirements and proposal nomenclature, they can facilitate partially decentralized orchestration for a wide range of information flows. Furthermore, the quorum-based DIV and DOV mechanisms can be tailored to meet the needs of different information flows that require decentralized decision-making, ensuring their broad applicability across various contexts.

D. Applicability, Limitations, and Future directions

DAO-FL, with its decentralized governance and validation mechanisms, is particularly suited for industries that prioritize the integrity of the GM and the FL process. Sectors like healthcare and finance, where privacy, transparency, and security are critical, stand to benefit greatly from DAO-FL's

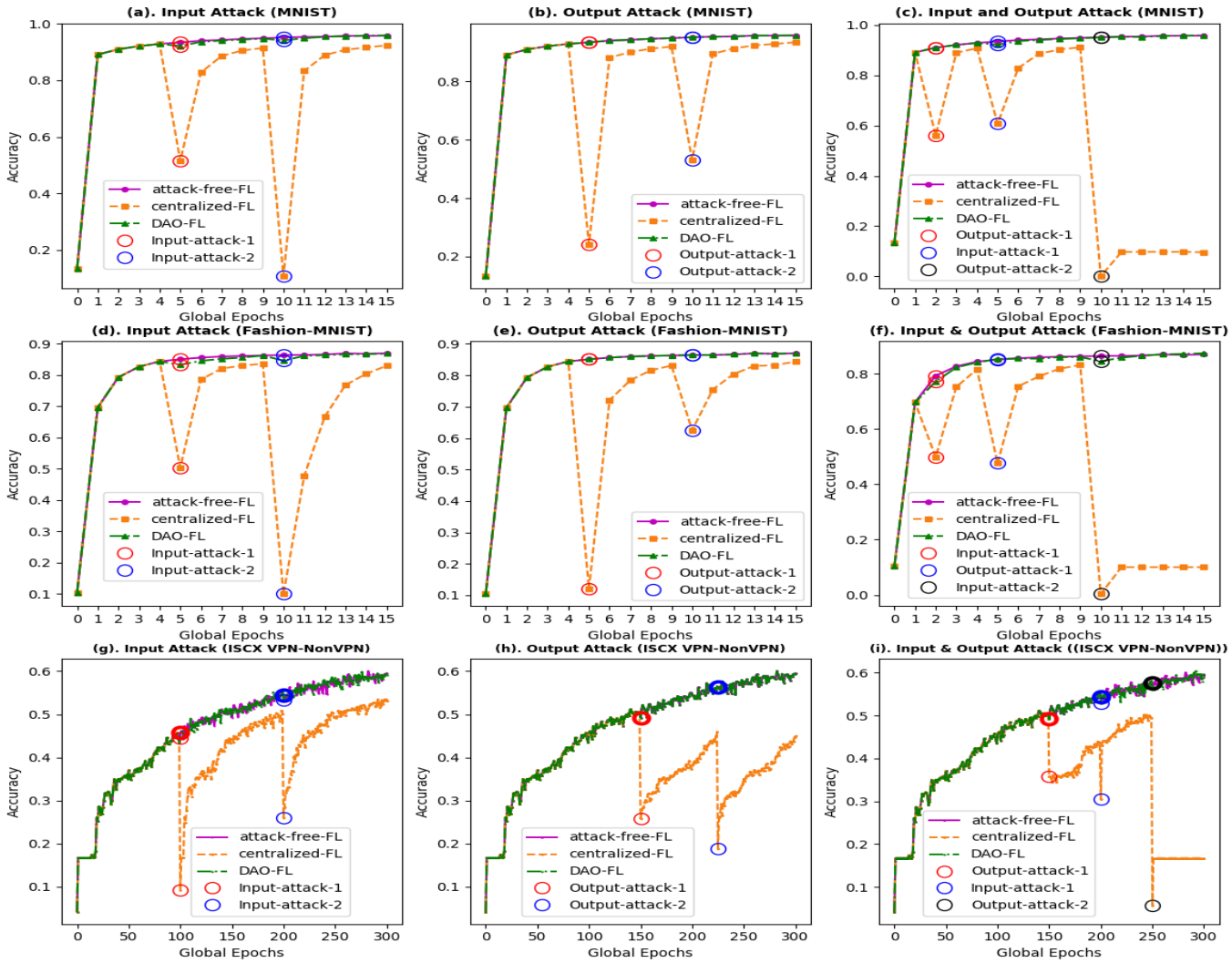

Fig. 17. Threat Evaluation of Input, Output, and Input & Output Attacks on DAO-FL, and centralized-FL (N=10, E=10).

decentralized approach, which ensures compliance with regulations and safeguards AI systems. In industries such as supply chain management and logistics, DAO-FL facilitates seamless collaboration among stakeholders while maintaining GM integrity. While DAO-FL prioritizes AI model security over rapid learning and low costs, its advantages in data integrity make it ideal for sectors where trust is paramount. In high-risk industries like critical infrastructure or defense, DAO-FL's decentralized validation mechanisms reduce cyber attack risks and improve system resilience, making it a valuable tool for protecting intellectual property and ensuring reliable machine learning models.

However, DAO-FL has limitations that hinder its widespread adoption, particularly in real-time and time-sensitive applications. Industries like high-frequency trading or autonomous vehicles, which demand low-latency decision-making, may find DAO-FL less applicable due to its complexity and higher transaction costs. Non-deterministic response times, reliance on blockchain network constraints, complex validation processes, and data transfer overheads pose significant challenges for rapid decision-making. Additionally, the decentralized nature of DAO-FL introduces variability in response times, limiting its suitability for real-time FL applications. Addressing these issues requires further research and optimization.

The cost analysis of DAO-FL is complex, influenced by factors such as gas fees, network congestion, and transaction volume. Table IV compares average transaction costs for FL-Incentivizer (centralized) and DAO-FL (decentralized) for IV and OV on both public (Sepolia) and private blockchains. On public blockchains, the transaction costs for decentralized IV and OV are higher than centralized IV and OV. Although private blockchains usually have no transaction costs, there is cost for establishing and running the network infrastructure. Furthermore, transforming a GM into an FL-NFT for commercial use may not be feasible on a private blockchain. High transaction costs of DAO-driven solutions on public blockchains, especially due to on-chain voting mechanisms, remain a challenge.

To overcome these limitations, several strategies can be explored. Off-chain voting mechanisms, such as those used by platforms like Snapshot and Aragon, can reduce on-chain transaction costs for DIV and DOV. Additionally, gas optimization techniques and layer-2 scaling solutions, like state channels or Plasma, can help reduce transaction fees

and alleviate blockchain congestion. Streamlining validation processes, minimizing data transfer overhead, and refining consensus mechanisms will improve DAO-FL's efficiency and responsiveness, especially in real-time applications. By addressing these challenges, DAO-FL can reach its full potential across diverse industries.

E. Case Studies

These case studies illustrate how DAO-FL effectively prevents and responds to input and output attacks in FL across various real-world scenarios.

1) Inventory and Logistics Operations in Supply Chain Management: In the evolving field of supply chain management (SCM), DAO-FL serves as a transformative solution that integrates DIV and DOV mechanisms to strengthen FL processes. In this scenario, stakeholders in a global supply chain collaborate in an FL setup to optimize inventory management and streamline logistics. DAO-FL allows these stakeholders to securely share LMs for collaborative training, ensuring the authenticity and integrity of inputs through decentralized validation. Malicious actors trying to inject incorrect LMs or manipulate the GM face significant barriers, as DAO-FL's robust verification protocols swiftly detect and counteract potential attacks. This safeguards the accuracy of SCM predictions, preserves the integrity of decision-making processes, and enhances operational efficiency and resilience within the supply chain ecosystem.

2) Fraud Detection in Financial Institutions: In financial institutions, FL systems play a crucial role in detecting fraud while maintaining customer confidentiality. To enhance fraud detection within privacy regulations, banks can collaborate under regulatory oversight, such as state banks, utilizing DAO-FL for joint fraud detection. At the end of a specified period, participating banks train their LMs on the latest transaction data and securely share these models per DAO-FL guidelines. The (GM is generated from the aggregated LMs. However, malicious actors might submit incorrect LMs or execute output attacks on the GM. By employing decentralized mechanisms like DIV and DOV to verify the reliability of LMs and GM updates, DAO-FL reinforces fraud prevention. This proactive strategy effectively detects and responds to fraudulent activities, safeguarding customer interests and ensuring the integrity of financial transactions.

VI. Conclusion

This article presents the DAO-FL framework, a groundbreaking approach to decentralized autonomous organizations to enhance FL processes. By incorporating decentralized input and output verification mechanisms, DAO-FL ensures the integrity and security of the FL ecosystem. Using DAO Membership Tokens (DAOMTs) and smart contracts like MultiSigC and DAOFLC demonstrates the framework's adaptability and versatility. With decentralized governance structures involving various stakeholders, DAO-FL provides a transparent and democratic framework for managing FL processes. Qualitative evaluations under different threat models showcase DAO-FL's superiority over traditional centralized FL approaches, particularly in scenarios requiring decentralized verification. Discussions on applicability across industries, transaction costs, and future directions highlight the framework's potential impact and scalability. Ultimately, DAO-FL strengthens FL integrity through decentralized decision-making and validation mechanisms, setting a new standard for decentralized orchestration in information flows. The DAO-FL framework's emphasis on security and decentralization, through multi-signature contracts and DAOs, encourages future research into enhanced security protocols in FL and decentralized systems, fostering resilient frameworks to mitigate centralization risks and build trust in collaborative data-sharing environments.

Appendix A
Demonstrative metadata for FL-NFT, ODAOMT, and VDAOMT

- Explore the FL-NFT's metadata at https://ipfs.io/ipfs/QmaCtmSJZrYXt9BQtZfk62zo5wzsQWW4ZpeF9cJ5USQFWE.
- Explore the metadata of ODAOMT at https://ipfs.io/ipfs/QmNPqQqiC1dwADZ2FLwtUi2nGi5CdkYxzZNEaroc3ZUS7R.
- Explore the metadata of VDAOMT at https://ipfs.io/ipfs/QmRrHTzcCJvFDWVq9DUnUTgxnCNyWUAANy8TyMRMeQhPp3.

Appendix B
UML Diagram for DAOMTC and DAOC

See the UML diagram at https://github.com/umermajeedkhu/DAOFLcode/blob/main/UML/appendixB.pdf.

Appendix C
UML Diagram for ODAOMTC, ODAOC, VDAOMTC, VDAOC, FLTokenC, DAOFLC, FLNFTC, and MultiSigC

See the UML diagram at https://github.com/umermajeedkhu/DAOFLcode/blob/main/UML/appendixC.pdf.

Appendix D
Inheritance graph of the DAOC, DAOMTC, ODAOC, VDAOC, ODAOMTC, VDAOMTC, and FLNFTC.

See the inheritance graph at https://github.com/umermajeedkhu/DAOFLcode/blob/main/graphs/appendixD.pdf.

Appendix E
Inheritance graph of the DAOFLC, MultiSigC, and FLTokenC.

See the inheritance graph at https://github.com/umermajeedkhu/DAOFLcode/blob/main/graphs/appendixE.pdf.

References

[1] 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, p. 103007, May 2021.

[2] R. Qin, W. Ding, J. Li, S. Guan, G. Wang, Y. Ren, and Z. Qu, "Web3-Based Decentralized Autonomous Organizations and Operations: Architectures, Models, and Mechanisms," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 53, no. 4, pp. 2073–2082, Apr. 2023.

[3] 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, Nov. 2019.

[4] 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.

[5] S. Wang, W. Ding, J. Li, Y. Yuan, L. Ouyang, and F.-Y. Wang, "Decentralized Autonomous Organizations: Concept, Model, and Applications," *IEEE Transactions on Computational Social Systems*, vol. 6, no. 5, pp. 870–878, Oct. 2019.

[6] E. G. Weyl, P. Ohlhaver, and V. Buterin, "Decentralized Society: Finding Web3's Soul," *Available at SSRN 4105763*, May 2022. [Online]. Available: https://ssrn.com/abstract=4105763

[7] T. J. Chaffer and J. Goldston, "On the Existential Basis of Self-Sovereign Identity and Soulbound Tokens: An Examination of the "Self" in the Age of Web3," *Journal of Strategic Innovation and Sustainability*, vol. 17, no. 3, Nov. 2022.

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

[9] A. Musamih, I. Yaqoob, K. Salah, R. Jayaraman, M. Omar, and S. Ellahham, "Using NFTs for Product Management, Digital Certification, Trading, and Delivery in the Healthcare Supply Chain," *IEEE Transactions on Engineering Management*, vol. 71, pp. 4480–4501, Nov. 2024.

[10] 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*, Fort Lauderdale, FL, USA, Apr. 2017, pp. 1273–1282.

[11] Z. Qin, J. Ye, J. Meng, B. Lu, and L. Wang, "Privacy-Preserving Blockchain-Based Federated Learning for Marine Internet of Things," *IEEE Transactions on Computational Social Systems*, vol. 9, no. 1, pp. 159–173, Feb. 2022.

[12] 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.

[13] T. Zeng, O. Semiari, M. Chen, W. Saad, and M. Bennis, "Federated Learning on the Road Autonomous Controller Design for Connected and Autonomous Vehicles," *IEEE Transactions on Wireless Communications*, vol. 21, no. 12, pp. 10 407–10 423, Dec. 2022.

[14] A. Trask, E. Bluemke, B. Garfinkel, C. G. Cuervas-Mons, and A. Dafoe, "Beyond Privacy Trade-offs with Structured Transparency," Dec. 2020, arXiv:2012.08347.

[15] 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.

[16] N. Z. Aitzhan and D. Svetinovic, "Security and Privacy in Decentralized Energy Trading Through Multi-Signatures, Blockchain and Anonymous Messaging Streams," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 5, pp. 840–852, Oct. 2018.

[17] U. Majeed, L. U. Khan, S. S. Hassan, Z. Han, and C. S. Hong, "FL-Incentivizer: FL-NFT and FL-Tokens for Federated Learning Model Trading and Training," *IEEE Access*, vol. 11, pp. 4381–4399, Jan. 2023.

[18] M. Adi Paramartha Putra, N. Bogi Aditya Karna, R. Naufal Alief, A. Zainudin, D.-S. Kim, J.-M. Lee, and G. A. Sampedro, "PureFed: An Efficient Collaborative and Trustworthy Federated Learning Framework Based on Blockchain Network," *IEEE Access*, vol. 12, pp. 82 413–82 426, Jun. 2024.

[19] A. Wahrstätter, S. Khan, and D. Svetinovic, "OpenFL: A scalable and secure decentralized federated learning system on the Ethereum blockchain," *Internet of Things*, vol. 26, p. 101174, Mar 2024.

[20] A. P. Kalapaaking, I. Khalil, X. Yi, K.-Y. Lam, G.-B. Huang, and N. Wang, "Auditable and Verifiable Federated Learning Based on Blockchain-Enabled Decentralization," *IEEE Transactions on Neural Networks and Learning Systems*, 2024, In press.

[21] E. Bluemke, T. Collins, B. Garfinkel, and A. Trask, "Exploring the Relevance of Data Privacy-Enhancing Technologies for AI Governance Use Cases," Mar. 2023, arXiv:2303.08956.

[22] M. I. Lunesu, R. Tonelli, A. Pinna, and S. Sansoni, "Soulbound Token for Covid-19 Vaccination Certification," in *IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*, Atlanta, GA, USA, Mar. 2023, pp. 243–248.

[23] U. Tejashwin, S. J. Kennith, R. Manivel, K. C. Shruthi, and M. Nirmala, "Decentralized Society: Student's Soul Using Soulbound Tokens," in *International Conference for Advancement in Technology (ICONAT)*, Goa, India, Jan. 2023.

[24] N. Diallo, W. Shi, L. Xu, Z. Gao, L. Chen, Y. Lu, N. Shah, L. Carranco, T.-C. Le, A. B. Surez, and G. Turner, "eGov-DAO: a Better Government using Blockchain based Decentralized Autonomous Organization," in *2018 International Conference on eDemocracy & eGovernment (ICEDEG)*, Ambato, Ecuador, Apr. 2018, pp. 166–171.

[25] P. Zhang, X. Hua, and H. Zhu, "Cross-Chain Digital Asset System for Secure Trading and Payment," *IEEE Transactions on Computational Social Systems*, vol. 11, no. 2, pp. 1654–1666, Feb. 2024.

[26] W. Ding, J. Hou, J. Li, C. Guo, J. Qin, R. Kozma, and F.-Y. Wang, "DeSci Based on Web3 and DAO: A Comprehensive Overview and Reference Model," *IEEE Transactions on Computational Social Systems*, vol. 9, no. 5, pp. 1563–1573, Oct. 2022.

[27] Y. Xiao, P. Zhang, and Y. Liu, "Secure and efficient multi-signature schemes for fabric: An enterprise blockchain platform," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 1782–1794, Dec. 2021.

[28] T. Chen, Z. Li, X. Luo, X. Wang, T. Wang, Z. He, K. Fang, Y. Zhang, H. Zhu, H. Li, Y. Cheng, and X. Zhang, "SigRec: Automatic Recovery of Function Signatures in Smart Contracts," *IEEE Transactions on Software Engineering*, vol. 48, no. 8, pp. 3066–3086, May 2022.

[29] R. Modi, *Solidity Programming Essentials: A guide to building smart contracts and tokens using the widely used Solidity language*. Packt Publishing, 2022.

[30] "Surya, The Sun God: A Solidity Inspector," Accessed: July 20, 2023. [Online]. Available: https://github.com/ConsenSys/surya

[31] Ethereum Magicians forum, "EIP-4671: Non-tradable Token," Accessed: July 20, 2023. [Online]. Available: https://ethereum-magicians.org/t/eip-4671-non-tradable-token/7976

[32] "OpenZeppelin ownable implementation," Accessed: July 20, 2023. [Online]. Available: https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/access/Ownable.sol

[33] "OpenZeppelin ERC721 implementation," Accessed: July 20, 2023. [Online]. Available: https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/token/ERC721/ERC721.sol

[34] "OpenZeppelin: The premier crypto cybersecurity technology and services company," Accessed: July 20, 2023. [Online]. Available: https://openzeppelin.com

[35] "OpenZeppelin ERC20 implementation," Accessed: July 20, 2023. [Online]. Available: https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/token/ERC20/ERC20.sol

[36] Nomic Foundation, "Hardhat - Ethereum development environment for professionals," 2021, Accessed: July 20, 2023. [Online]. Available: https://hardhat.org/

[37] "Sepolia Test Network," Accessed: July 20, 2023. [Online]. Available: https://sepolia.etherscan.io/

[38] Etherscan, "Etherscan APIs," Accessed: July 20, 2023. [Online]. Available: https://etherscan.io/apis

[39] G. Draper-Gil, A. H. Lashkari, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of Encrypted and VPN Traffic using Time-related Features," in *Proceedings of the 2nd International Conference on Information Systems Security and Privacy (ICISSP)*, Rome, Italy, 2016, pp. 407–414.

[40] U. Majeed, L. U. Khan, and C. S. Hong, "Cross-Silo Horizontal Federated Learning for Flow-based Time-related-Features Oriented Traffic Classification," in *21st Asia-Pacific Network Operations and Management Symposium (APNOMS)*, Daegu, Korea (South), Oct. 2020, pp. 389–392.

Umer Majeed received his BS degree in Electrical Engineering from the National University of Science and Technology, Pakistan, in 2015. He is currently pursuing his Ph.D. degree in Computer Science and Engineering at Kyung Hee University, South Korea. Mr. Majeed has served as the Technical Program Committee Member of the Edge Intelligence for Internet of Things Workshop during the 20th International Conference on Wireless Communications and Mobile Computing (IWCMC), 2024. His research interests span a wide spectrum, encompassing Blockchain, Web3, Smart Contracts, Non-Fungible Tokens, Decentralized Autonomous Organizations, Structured Transparency, Private & Secure AI, Internet of Things, Edge Computing & Intelligence, Smart Cities, and Federated Learning.

Sheikh Salman Hassan (S'14, M'24) received his B.S. degree in Electrical Engineering with magna cum laude honors from the National University of Computer and Emerging Sciences, Pakistan, in 2017 and the Ph.D. degree in Computer Science and Engineering from Kyung Hee University, Seoul, South Korea, in 2024. Currently, Dr. Hassan holds a Postdoctoral Research Fellowship position with the Networking Intelligence Lab at Kyung Hee University. Dr. Hassan has served as the Technical Program Committee (TPC) Member of the Edge Intelligence for Internet of Things Workshop during the 20th International Conference on Wireless Communications and Mobile Computing (IWCMC), 2024. His research interests lie in the areas of 6G networks, non-terrestrial networks (NTNs), the Internet of Everything (IoE), network resource management, intelligent networking, and the application of game theory, optimization theory, and machine learning to network problems. Dr. Hassan's achievements include Best Poster and Best Oral Paper awards received at the International Conference on Information Networking (ICOIN) in 2021 and 2023, respectively

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-F'24) 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.