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. The lack of transparency and security in traditional FL systems can be attributed to the centralized validation of LMs and GM updates. In this paper, we propose DAO-FL, a smart contract-based framework that leverages the power of DAOs to address these FL challenges. 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 employs a multi-signatures approach facilitated by an Orchestrator-DAO to achieve decentralized GM updates, and thus 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 demands days of training for the initial deployable GM and continuous updates over extended periods. In traditional "centralized FL" setups, LMs are validated by a central server, which aggregates them to produce the GM. However, this centralized approach poses vulnerabilities, as a single erroneous GM update can potentially compromise the accuracy and integrity of the entire FL process. To tackle these challenges, in this study, we propose the DAO-FL framework, which integrates DAOs and a multi-signature [16] contract with FL to enable DIV and DOV of the FL process. By employing DAOs, we distribute

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.

the verification process across multiple participants, ensuring transparency and mitigating the risk of central authority manipulation. 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.

TABLE I
LIST OF ABBREVIATIONS, SYMBOLS AND DESCRIPTION

Abbreviation	Symbol	Description
CID	-	Content Identifier
DIV	-	Decentralized input verification
DOV	-	Decentralized output verification
DAO	-	Decentralized Autonomous Organization
DAOC	-	DAO contract
DAOFLC	-	DAO-FL contract
DAOMT	-	DAO Membership Token
FL	-	Federated Learning
FLNFTC	-	FL-NFT contract
FLT	FLT	FL Task
FLTP	-	FL-task publisher
FLTokenC	-	FL Token contract
GI	t	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	-	local model
LMUs	-	local model uploads
LMURI	-	local model Uniform Resource Identifier
LMCID	-	local model (IPFS) Content Identifier
MultiSigC	-	Multi-Signature contract
NFT	-	Non-fungible Token
NTT	-	Non-transferable Token
ODAO	-	Orchestrator-DAO
ODAOC	-	Orchestrator-DAO contract
ODAOM	$ODAOM_i$	Orchestrator-DAO member
ODAOMT	-	Orchestrator-DAO Membership Token
ODAOMTC	-	Orchestrator-DAOMT contract
OV	-	Output verification
SBT	-	Soul-Bound Token
ST	-	Structured Transparency
URI	-	Uniform Resource Identifier
VDAO	-	Validation-DAO
VDAOC	-	VDAO contract
VDAOM	$VDAOM_i$	VDAO member
VDAOMT	-	Validation-DAO Membership Token
VDAOMTC	-	Validation-DAOMT contract

II. RELATED WORK

Bluemke *et al.* in [18] explored the significance of data privacy-enhancing technologies in the realm of AI governance. They highlighted the progress made in balancing privacy and performance during data exchange and analysis, emphasizing the value of ST. Thus, enabling controlled information flow, addressing who, when, and how information should be accessible, and ensuring efficient collaboration while reducing data misuse risks.

Majeed *et al.* in [1] proposed the ST-BFL framework, utilizing homomorphic encryption, FL-aggregators, FL-verifiers, and a smart contract to satiate components of ST [14] for FL process. Homomorphic encryption ensures input privacy, and FL-verifiers validate the GM for OV. However, ST-BFL lacks LM validation as it prioritizes input privacy over IV. Additionally, detailed information on authentication and authorization of FL-verifiers, vital for OV, is missing. In contrast, DAO-FL focuses on DAO-based IV and OV of the FL process. Majeed *et al.* proposed FL-Incentivizer in [17], incentivizing device participation in FL with FL-Tokens and enabling ownership rights to a GM via FL-NFT. FL-Incentivizer employs an FLTPCO for LMs' validation and GM updates, ensuring IV and OV centrally. However, this work extends FL-Incentivizer by decentralizing the IV and OV processes through DAOs and a multi-signature contract. Table. II compares the ST

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

TABLE II
MAPPING OF STRUCTURED TRANSPARENCY TO ST-BFL, FL-INCENTIVIZER, AND DAO-FL

ST Component	ST-BFL [15]	FL-Incentivizer [17]	DAO-FL (This Work)
Input Privacy	• Input privacy is maintained by employing homomorphic encryption to encrypt all the LMs.	• Input privacy is ensured by relying on the self-capabilities of FL, where LMs are sent to the server instead of raw data.	• DAO-FL achieves input privacy by leveraging on self-capability of FL where LMs, instead of raw data, are transmitted to the server.
Output Privacy	• Output privacy is maintained during the aggregation process by producing a homomorphically encrypted GM. The decryption of the GM is restricted to the FLTP, ensuring GM's confidentiality.	• Output privacy in FL-Incentivizer is guaranteed by the self-capabilities of FL, where the aggregated GM prevents the leakage of LM privacy.	• Output privacy in DAO-FL is guaranteed through the inherent capabilities of FL, as the aggregated GM prevents any potential privacy breaches associated with the LMs.
Input Verification	• In ST-BFL, IV is a challenging aspect to achieve alongside input privacy. The current research indicates that simultaneous attainment of IV and input privacy is difficult.	• The IV process in FL-Incentivizer is centralized, with FLTPCO being responsible for validating and approving the LMs submitted by participating devices.	• DIV is accomplished by the VDAO utilizing DAO-based voting mechanism to validate LMs.
Output Verification	• ST-BFL framework employs an FL-aggregator to generate the output, which is the GM, by aggregating the LMs. • To ensure the accuracy and reliability of the generated GM, FL-verifiers participate by voting on whether the FL-aggregator has aggregated the LMs correctly.	• FLTPCO, as a central authority in FL-Incentivizer, is responsible for generating the updated GM as output and maintaining a record of it within the FLTPC contract, making FLTPCO solely accountable for OV.	• The potential updated GM is generated by the FLTP and put forward in "GM Update" proposal within a multi-signature contract for approval by ODAOMs. • The DOV is accomplished through a voting mechanism within the multiSigC, which is facilitated by the ODAO.
Flow Governance	• ST-BFL framework incorporates flow governance by utilizing a smart contract and entities such as ST-BFL market service manager, FLTP, FL-aggregators, FL-verifiers, and FL-trainers.	• Flow governance in FL-Incentivizer is upheld by smart contracts and FLTP. • FL-Incentivizer enables participant incentivization through FLTokens and the tokenization of the GM as an NFT.	• The flow governance is maintained by various smart contracts as well as FLTP, ODAOMs, and VDAOMs. • DAO-FL empowers ODAOMs, and VDAOMs to perform member enrollment and member expulsion operations enabling decentralized flow governance.

components of ST-BFL, FL-Incentivizer, and the proposed work "DAO-FL".

Lunesu *et al.* in [19] presented a practical application of SBTs for COVID-19 vaccine certification using the decentralized Vaccine System DApp, powered by blockchain. The research explains system components, smart contracts, user interface, and database, while also addressing the roles and actions of citizens and administrators within the system. It emphasizes the potential of SBTs in establishing a reliable decentralized society, and self-sovereign identity (SSI). They also discuss associated challenges and privacy concerns. [20] proposed an innovative approach that utilizes SBTs to encode individuals' affiliations and academic credentials in a decentralized network. The system employs off-chain storage, smart contracts, and cryptographic technologies to enhance privacy and security, and offers a trustworthy environment for stakeholders, providing a robust and confidential alternative to centralized academic credential verification.

Diallo *et al.* in [21] presented an eGov-DAO system to enhance e-government transaction efficiency, transparency, and security. Through the implementation of a DAO and smart contracts, the system automates transactions, thereby 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. It protects against theft by requiring multiple signatures for validity. This approach fosters a secure and trustworthy energy trading system without reliance on trusted third parties, promoting a more decentralized and competitive environment for energy trade.

III. PRELIMINARIES

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

A. Decentralized Autonomous Organization

A DAO [22] is an internet-native digital equivalent to traditional companies in the physical world. DAOs, in essence, allow members to create and vote on governance decisions that are specifically made by the boards of directors or executives in conventional companies. A DAO operates autonomously following predefined business logic contained in its smart contract to accomplish a collective mission of DAO's community with token economy-based incentives. "The DAO," launched in 2016, was the world's first DAO and raised $150 million in Ether (ETH), making it one of the largest digital crowdfunding projects. Some other popular examples of DAOs are DigixDAO, Aragon, Steemit, etc. DAO has an initial creation phase in which typically EOAs send Ethers to the DOA smart contract's address and DOA tokens are created and assigned to those EOAs as proof of DOA's membership and voting rights. DAOs make it possible to accomplish a broad spectrum of objectives, encompassing activities such as delivering services, generating targeted funds, owning and managing smart assets, coordinating with other autonomous software, and facilitating cooperation among various stakeholders.

B. Structured Transparency

Structured transparency [14] is a framework designed to address the tradeoff between privacy and transparency for information flows. It consists of five components: input privacy, output privacy, IV, OV, and flow governance. Input privacy ensures that confidential information can be processed without being disclosed to unauthorized parties. On the other hand, output privacy enables individuals to participate in information flows and contribute data without the risk of sensitive input being exposed in the resulting output. IV involves ensuring the integrity of the input, while OV ensures that the output has not been tampered with. Flow governance refers to the overall management and control of the information flow. To satisfy each component, certain requirements must be met. Input privacy requires mechanisms to process information without revealing it, while output privacy necessitates preventing the inference of sensitive input from the output. IV requires methods to ensure the integrity and authenticity of the input, and OV requires techniques to prove that the output has not been tampered with. Flow governance requires effective management and control mechanisms to govern the entire information flow.

C. Multi-signature wallet

A multi-signature (also known as a "multisig") wallet is a type of digital wallet that enhances security by requiring more than one person to sign off on a transaction before it can be executed [23]. In multi-signature wallets, the execution of transactions is governed by the quorum quotient, which is represented by the m-of-n ratio. This ratio refers to the minimum number of signatories required to sign a transaction, expressed as a fraction of the total number of registered signatories. For instance, a 3-of-5 wallet mandates that at least three out of five designated signers must approve a transaction for it to be processed. This can be useful in cases where multiple parties need to agree on a transaction, or where added security is desired to protect against unauthorized transactions. Multi-signature wallets are commonly used in a variety of contexts, including financial transactions, corporate governance, and the management of cryptocurrency exchanges. Multi-signature wallets are commonly implemented using smart contracts to enforce the requirement of multiple signatures for transaction authorization.

IV. PROPOSED FRAMEWORK

This section offers a comprehensive explanation of the proposed system architecture and execution workflow within the DAO-FL framework. The system architecture, depicted in Fig. 1, comprises three blocks: the administrative block, the decentralized block, and the FL-trainer block.

The administrative block consists of pivotal stakeholders in the DAO-FL framework including a regulator, FL-task-publisher (FLTP), Orchestrator-DAO (ODAO), and Validation-DAO (VDAO). These entities govern and orchestrate various aspects of the DAO-FL ecosystem. The regulator governs the FL ecosystem, deploys the FLNFTC, and standardizes FLNFTs metadata. We denote the regulatory entity as $Regulator$. When an entity referred to as the FLTP adopts the DAO-FL framework to train an FL model, it must deploy specific smart contracts, namely ODAOC, VDAOC, DAOFLC, and MultiSigC, customized exclusively for the specific FL task. The ODAO, a DAO overseeing the FL process, comprises multiple members ($ODAOM_i$). These Orchestrator-DAO members (ODAOMs) are responsible for approving proposals from the FLTP and possess the ability to aggregate LMs. Similarly, the VDAO verifies the LMs submitted by FL-Trainers by utilizing its VDAO members (VDAOMs), where each $VDAOM_i$ can validate LMs relevant to the given FL task.

The decentralized block consists of essential 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), FL Token contract (FLTokenC), and InterPlanetary File System (IPFS). The FLNFTC, derived from the ERC-721 standard and deployed by the regulator, enables the tokenization of FLT's GM. ODAOC manages membership operations within the ODAO, while ODAOMTC mints Orchestrator-DAOMTs (ODAOMTs) for ODAOMs. Similarly, VDAOC handles member-related operations in the VDAO, and VDAOMTC generates Validation-DAOMTs (VDAOMTs) for VDAO members. A comprehensive explanation of DAOMTs is provided in Section IV-A. Both ODAOMTC and VDAOMTC are customizations of ERC-721 standard. It is worth noting that the ODAOMTC and VDAOMTC are deployed upon the deployment of the ODAOC and VDAOC respectively. The DAOFLC orchestrates the FL process for a given FL task, supported by MultiSigC for decentralized execution. The MultiSigC, in turn, facilitates the decentralized execution of FL operations within the DAOFLC by collecting multiple signatures from $ODAOM_i$. FLTokenC, deployed by DAOFLC, manages FL-Tokens specific to each FL task. IPFS serves as a decentralized file storage system for metadata, LMs, and GM.

The FL-Trainers block consists of multiple FL learners, with each FL-Trainer representing a participating device or client in the FL process. We denote the FL-Trainer for the

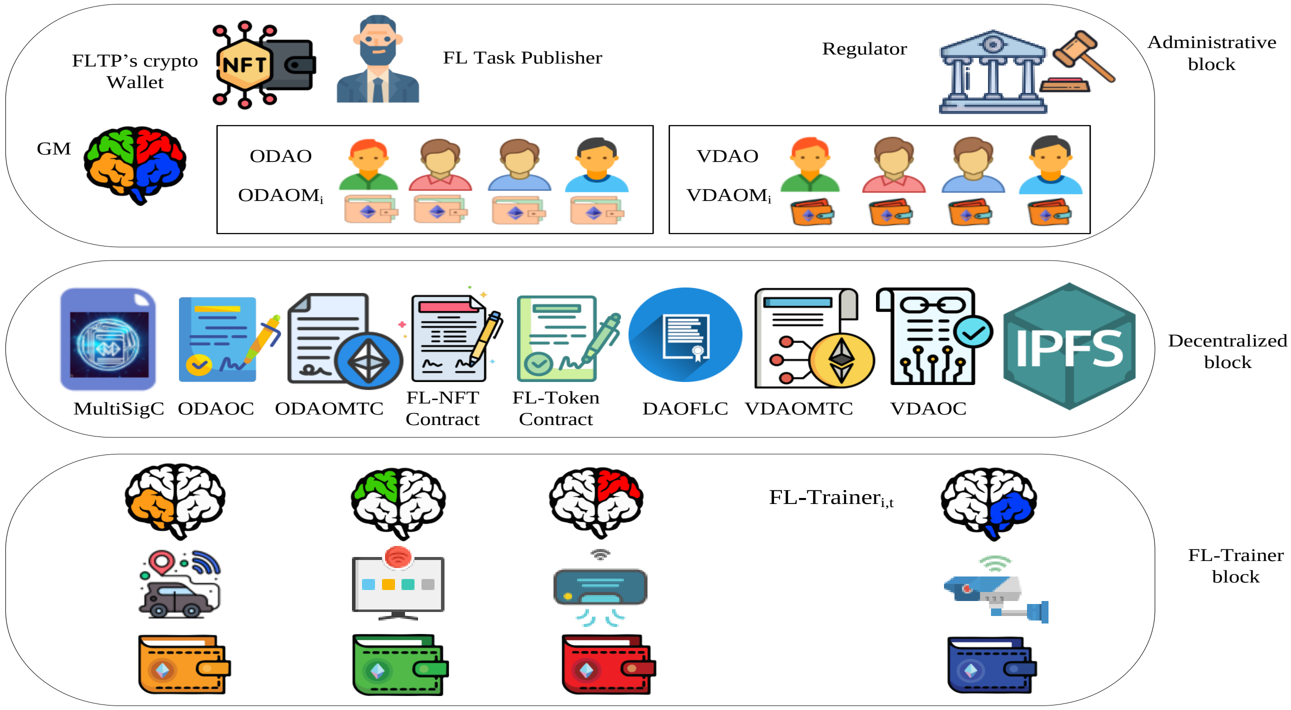

Fig. 1. DAO-FL: System Architecture.

i^{th} client in the $t+1^{th}$ generation interval of FL task as $FLTrainer_{i,t+1}$. The FL-Trainer retrieves and downloads the GM_{t+1} and generates its local model upload $LMU_{i,t+1}$ utilizing its respective local dataset $D_{i,t+1}$.

Besides the previously mentioned entities, the system architecture also includes two crucial components: FL-NFTs and FL-Tokens. Each FL-NFT, denoted as $FLNFT$, is an ERC-721 compliant dynamic NFT associated with an FL task. It possesses a distinct numeric identity, referred to as $FLNFTID$. The FL-NFT is equipped with a Uniform Resource Identifier (URI) called $tokenURI$ that links to the metadata of the current GM for the FL task [17]. Additionally, the FLNFT includes the $GMCID$ property, which represents the IPFS Content Identifier (CID) of the most recent GM. Crucially, the FL-NFT contains the address of the corresponding DAOFLC, known as $OrchestratorAddress$. The $tokenURI$, $GMCID$, and $OrchestratorAddress$ for each FL-NFT are distinctive. The FLTP, as the owner of the FLNFT, facilitates the benefits of GM commercialization and tokenization [17]. Furthermore, FL-Tokens, symbolized as $FLToken$, conform to the ERC-20 standard and are awarded to FL-Trainers within the FL process [17].

An overview of subsequent subsections is presented as follows: In Section IV-A, we introduce the novel concept of DAOMTs. Section IV-B proposes a member enrollment scheme for adding new members to a DAO, while Section IV-C presents a member expulsion scheme to address inactive or malicious members. Furthermore, Section IV-D outlines a mechanism for transferring ODAOC or VDAOC to a new proprietor. In Section IV-E, a scheme is proposed for partially decentralized orchestration of FL process in the DAOFLC using the MultiSigC. Additionally, Section IV-F details a comprehensive execution workflow for the DAO-FL framework, orchestrating the FL process from initial setup to completing a full GI. Lastly, Section IV-G delves into GM commercialization, involving the transfer of FL-NFT and contracts ownership to the new proprietor.

A. *DAO Membership Tokens (DAOMTs)*

DAOs are decentralized organizations that operate autonomously on a blockchain, governed by their members through a voting-based decision-making process. DAOMTs are a specific type of token designed to represent the membership of entities within a DAO. They are classified as NTTs and SBTs [6], meaning they cannot be traded or transferred on a marketplace. Additionally, DAOMTs are categorized as NFTs, with each token being unique. These tokens can be minted or burnt, denoting controlled creation and destruction, respectively. Typically, members are limited to holding one token per address, thereby restricting the maximum balance to one token per address. DAOMTs can be grouped together with other tokens to represent various levels or types of membership, forming a collection. They can be utilized for the governance of DAO-based systems, granting members the right to vote on proposals and participate in decision-making processes regarding the organization's direction and operation. Ultimately, DAOMTs contribute to a more democratic and decentralized approach to decision-making within a DAO.

B. *Membership Enrollment in ODAO and VDAO*

The process of becoming a member of ODAO or VDAO follows a similar procedure. Hence, in this section, we will describe the steps for joining a DAO through a DAO contract

Algorithm 1 : Membership Enrollment via DAOC

Caller: $DAOM_p$

procedure proposeJoin(**address** $candidate$)
  Ensure $DAOM_p$ **holds** a $DAOMT$
  if $candidate \notin DAO$ and $JoinProposals[candidate].open == false$ **then**
    Create ***new*** JP
    Set $JP.proposer$ = $DAOM_p$
    Set $JP.candidate$ = $candidate$
    Set $JP.open$ = $true$
    Set $JP.approvalvotes$ = $JP.denialvotes$ = 0
    Set $JP.voters$ = *empty* AddressSet
    Add JP to $JoinProposals$
  end if
end procedure

Caller: $DAOM_v$

procedure voteJoin(**address** $candidate$, **bool** V_v)
  Ensure $DAOM_v$ **holds** a $DAOMT$
  Set JP = $JoinProposals[candidate]$
  if $JP.open == true$ and $DAOM_v \notin JP.voters$ **then**
    if V_v==true **then**
      Add Approval vote for $DAOM_v$
    else
      Add Deny vote for $DAOM_v$
    end if
    Count $JP.approvalvotes$ and $JP.denialvotes$
    Q = 60% $* \, n(DAOMT)$
    if $JP.denialvotes > Q$ **then**
      Mint DOAMT for $candidate$
      Set $JP.open = false$
    else if $JP.denialvotes > Q$ **then**
      Set $JP.open = false$
    end if
  end if
end procedure

(DAOC), which is inherited by both ODAOC and VDAOC. After the creation of the DAO, it is essential to have pre-existing members. Let us denote the existing member within the DAO as $DAOM_i \in DAO$. The simplified sequential outline for joining a DAO is as below:

- Step 1: When a new $candidate$ seeks to join the DAO, a current member of the DAO, denoted as $DAOM_p$, will initiate a "proposeJoin" transaction to the DAOC. This transaction includes the candidate's address as an argument, effectively proposing its inclusion into the DAO.
- Step 2: To process the "proposeJoin" transaction, the DAOC first validates that the submitter ($DAOM_p$) possesses a DAOMT.
- Step 3: If the candidate is not a current member of DAO and no existing "Join Proposal" exists for it, a new "Join Proposal" (JP) is initiated. The JP includes the candidate's address and is proposed by $DAOM_p$. A boolean flag called "open" is set to $true$ to indicate that JP is currently being processed and has not been accepted or rejected. The $approvalvotes$ and $denialvotes$ fields of the JP are initialized to 0, indicating no approval or denial votes have been cast yet. The set of voters for the JP is initially empty, indicating no $DAOM_i \in DAO$ have voted for the JP yet.

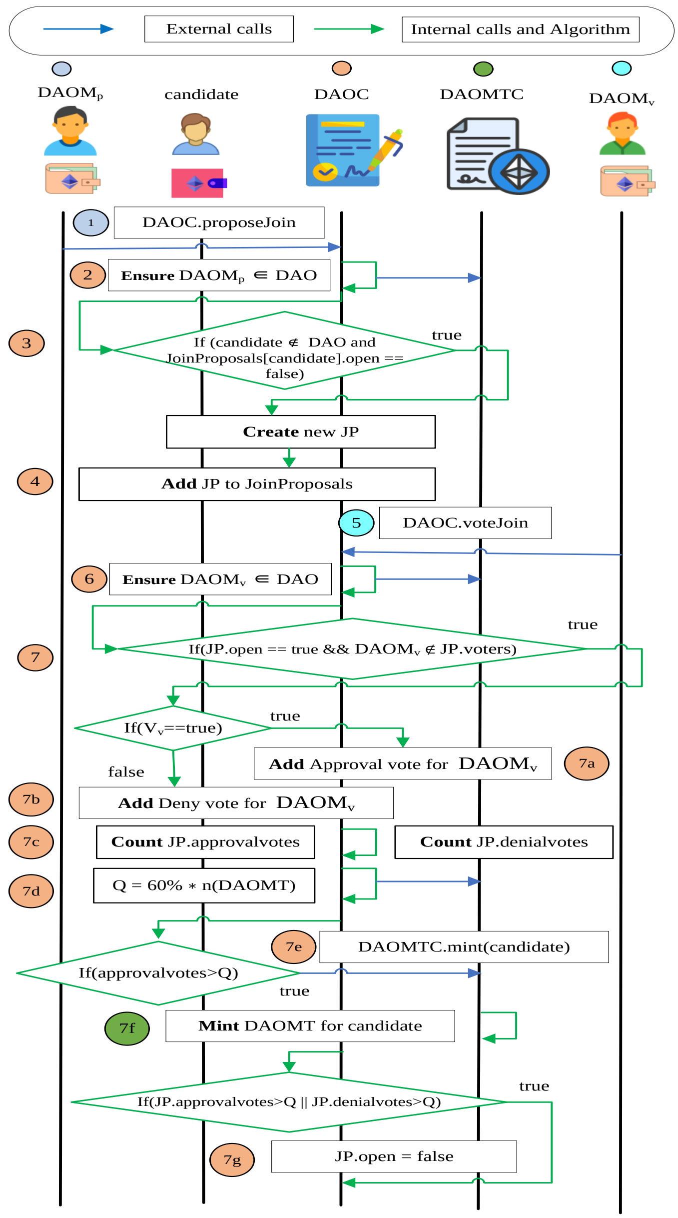

Fig. 2. Membership Enrollment in DAO - Sequence diagram.

- Step 4: Subsequently, the JP is then stored in a mapping data structure called the JoinProposals with $candidate$ as the index.

Steps 1-4 are combined in the procedure $proposeJoin$ (Algo. 1). The existing DAO members vote to accept or reject the JP. The voting procedure consists of the following steps:

- Step 5: When a $DAOM_v$ intends to vote on a JP, it will initiate a "voteJoin" transaction within $DAOC$, providing the $candidate$'s address and a boolean variable (V_v) representing their voting decision. The value "true" of V_v signifies the approval of $DAOM_v$ for the JP, while "false" indicates disapproval.
- Step 6: To prevent spam transactions, the $DAOC$ will first verify that the $DAOM_v$ possesses a valid $DAOMT$.
- Step 7: If an open JP exists for $candidate$ and $DAOM_i$ has not yet voted on it, their vote is added to the list $JP.voters$. 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, defined as $Q = 60\% * n(DAOMT)$, is 60% of the total supply of DAOMTC. If the $JP_{approvalvotes}$ surpasses Q, DAOC mints DAOMT for the $candidate$ through DAOMTC, closing JP by setting the "open" flag to false. Conversely, if $JP_{denialvotes}$ surpasses Q, the JP is rejected by setting the "open" flag 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 non-active or malicious members in a DAO raises concerns and calls for their expulsion. Non-active members fail to actively participate in the orchestration of the FL process, while malicious members engage in endorsing inaccurate updates. The procedure for removing members from both ODAO and VDAO is consistent, and a kick-out mechanism is introduced to address these non-active or malicious individuals. The simplified kick-out mechanism encompasses the following sequential steps:

- Step 1: When a DAO member, identified as $DAOM_p$, determines that another member (referred to as $candidate$) should be expelled, $DAOM_p$ initiates the kick-out process by submitting a "proposeKick" transaction to the DAOC. This transaction includes the address of the targeted $candidate$ as an argument.
- Step 2: $DAOC$ verifies if $DAOM_p$ holds a $DAOMT$ to prevent spam transactions.
- Step 3: If the $candidate$ is a member of the DAO and there is no existing "Kick Proposal" in progress for the $candidate$, a new "Kick Proposal" (KP) is initiated. The $candidate$ is specified as the target of the KP, and $DAOM_p$ assumes the role of the proposer. The KP is marked as "open" to indicate its ongoing status, awaiting acceptance or rejection. Initially, the KP has no approval or denial votes, so both $approvalvotes$ and $denialvotes$ fields of KP are set to zero. The set of voters for the $KP.voters$ is empty.
- Step 4: The KP is added to a mapping structure called KickProposals, with the $candidate$ serving as the index.

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

- Step 5: In the DAO's kick proposal voting process, a $DAOM_v$ can cast their votes through a transaction called "voteKick" to DAOC. It includes the $candidate$'s address and a boolean variable (V_v) indicating approval (true) or disapproval (false), as arguments.
- Step 6: The DAOC verifies that both $DAOM_v$ and $candidate$ hold a DAOMT.

Algorithm 2 : Member Expulsion via DAOC

Caller: $DAOM_p$
procedure proposeKick(**address** $candidate$)
  Ensure $DAOM_p$ **holds** a $DAOMT$
  if $candidate \in DAO$ and $KickProposals[candidate].open == false$ **then**
    Create ***new*** KP
    Set $KP.proposer$ = $DAOM_p$
    Set $KP.candidate$ = $candidate$
    Set $KP.open$ = $true$
    Set $KP.approvalvotes$ = $KP.denialvotes$ = 0
    Set $KP.voters$ = *empty* AddressSet
    Add KP to $KickProposals$
  end if
end procedure

Caller: $DAOM_v$
procedure voteKick(**address** $candidate$, **bool** V_v)
  Ensure $DAOM_v$ **holds** a $DAOMT$
  Ensure $candidate$ **holds** a $DAOMT$
  Set KP = $KickProposals[candidate]$
  if $KP.open == true$ and $DAOM_v \notin KP.voters$ **then**
    if V_v==true **then**
      Add Approval vote for $DAOM_v$
    else
      Add Deny vote for $DAOM_v$
    end if
    Count $KP.approvalvotes$ and $KP.denialvotes$
    Q = $60\% * n(DAOMT)$
    if $KP.approvalvotes > Q$ **then**
      Burn DOAMT owned by $candidate$
      Set $KP.open = false$
    else if $KP.denialvotes > Q$ **then**
      Set $KP.open = false$
    end if
  end if
end procedure

- Step 7: If the KP is open for a specific $candidate$ and $DAOM_v$ has not yet voted, their vote is added to the list $KP.voters$. 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

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

respectively. The quorum, defined as $Q = 60\% * n(DAOMT)$, is 60% of the total supply of DAOMTC. If the $KP_{approvalvotes}$ surpasses Q, DAOC burns the DAOMT owned by the $candidate$, closing KP by setting the "open" flag to false. Conversely, if $KP_{denialvotes}$ surpasses Q, the KP is rejected by setting the "open" flag to false.

Steps 5-7, for a KP, are summarized in procedure $voteKick$ (Algo. 2). The sequential flow for kicking out a DAO's member is depicted in Fig. 3.

D. Transferring ODAOC and VDAOC

The FLTP owns the GM, which is authenticated through the corresponding FL-NFT in FLNFTC. Additionally, the FLTP also holds ownership of ODAOC and VDOAC. When

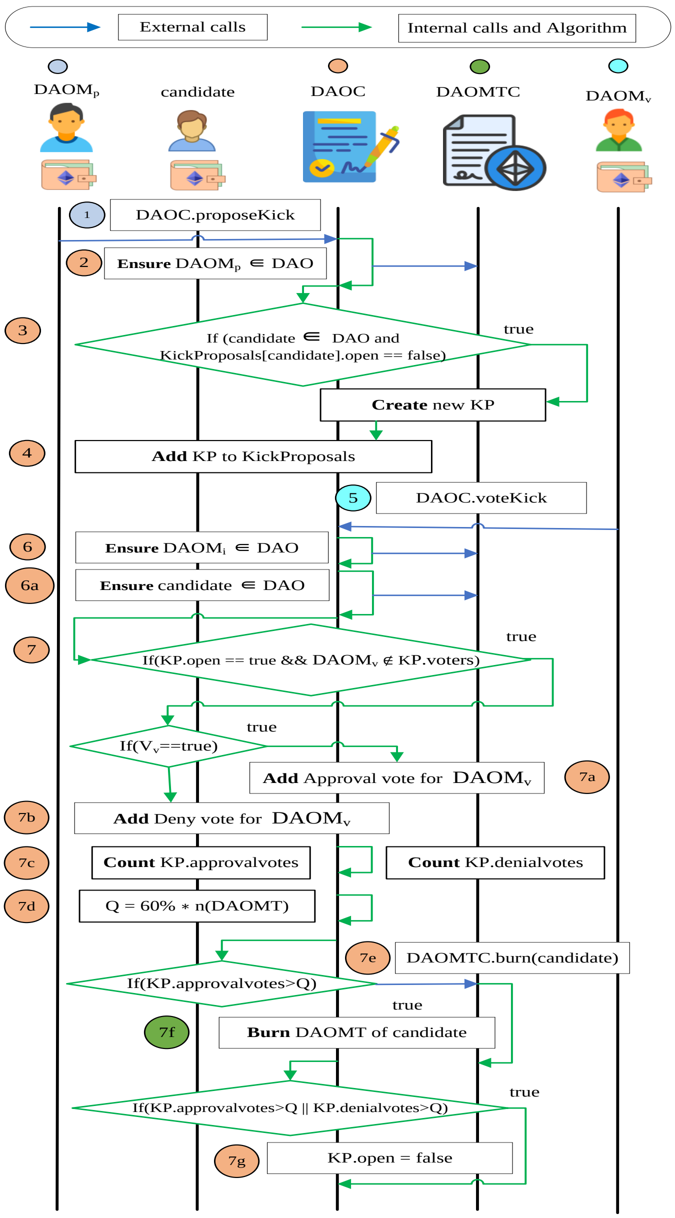

Fig. 3. Member Expulsion from DAO -Sequence diagram.

transferring ownership of the FLNFT, ownership of ODAOC and VDOAC must be transferred to the successor proprietor. The steps for transferring ownership of the DAOC, the parent contract of ODAOC and VDAOC, as outlined in the procedure $transferOwnership$ (Algo. 3), are as follows:

- Step 1: The current owner (FLTP) initiates a "transfer ownership" transaction to DAOC with the address of the new owner ($newOwner$) as an argument.
- Step 2: The DAOC verifies that the new owner $newOwner$ is different from the previous owner, and proceeds to transfer ownership of DAOC to $newOwner$. If $newOwner$ is not already a member of the DAO, a DAOMT is minted for $newOwner$, while the DAOMT owned by $oldOwner$ is burned to maintain scarcity.

Algorithm 3 : Transferring $DAOC$

```
Caller: FLTP        Modifier: onlyOwner()
procedure transferOwnership(address newOwner)
    oldOwner = owner()
    if oldOwner != newOwner then
        Transfer ownership of DOAC to newOwner
        if newOwner ∉ DAO then
            Mint DAOMT for newOwner
            Burn DAOMT of oldOwner
        end if
    end if
end procedure
```

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

In a multi-signature wallet setup, a Multi-Signature Contract (MultiSigC) is utilized to gather necessary signatures or votes from specified individuals for a transaction. Upon reaching the required quorum, the MultiSigC executes the transaction within the designated contract. In DAO-FL, the MultiSigC consolidates votes from ODAOMs to facilitate decentralized approval on different proposals to execute corresponding transactions in the DAOFLC, aiding in the orchestration of the FL process. It is important to highlight that while the MultiSigC handles this decentralized approval, the FLTP retains sole responsibility for executing approved proposals, resulting in a partially decentralized orchestration process. 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 covers various proposals like "createFLNFT", "Initiate_LMUs", "Cease_LMUs", "setLMUVDRF", or "UpdateGM". After verifying the submitter's identity, MultiSigC rigorously validates the transaction based on arguments, proposal type, and current state. Upon successful validation, a new "Proposal" is created with a unique identifier ($proposalID$) and set to "Open" state. The proposal's selector is configured using the corresponding function signature within the DAOFLC. Subsequently, the FLTP engages off-chain to secure ODAOMs' approval. This step is encapsulated in the procedure $propose$ (Algo. 4).
- Step 2: ODAOMs validate the proposal off-chain, considering its properties, nature, and the states of MultiSigC and DAOFLC. If valid, an $ODAOM_v$ initiates an "approve" transaction towards MultiSigC, including the $proposalID$ as an argument. MultiSigC verifies the transaction's legitimacy, checks if the proposal is open, and confirms that the $ODAOM_v$ has not voted previously. MultiSigC rigorously validates proposals based on relevant arguments, proposal's nature, and the current state. Upon successful validation, an 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 state is updated to the

Algorithm 4 : MultiSigC

```
Caller: FLTP      Modifier: onlyOwner()
procedure propose([selector], [tokenURI], [GMCID], [t + 1])
    Require:Caller==MultiSigC.owner()
    Validate propose
    if propose is valid then
        proposal = Create new Proposal with proposalID
        Set proposal.state = Open,   Set proposal.selector
    end if
end procedure
procedure execute(uint proposalID)
    Require:Caller==MultiSigC.owner()
    state = Proposal[proposalID].state
    if 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
    end if
end procedure
procedure closeProposal(uint proposalID)
    Require:Caller==MultiSigC.owner()
    state = Proposal[proposalID].state
    if state == Open or state == Executable then
        Set Proposal[proposalID].state = Closed
    end if
end procedure

Caller: ODAOM_v
procedure approve(uint proposalID)
    Ensure ODAOM_v holds a ODAOMT
    proposal = Proposal[proposalID]
    if proposal.state == Open and ODAOM_v ∉ proposal.approvals then
        Add ODAOM_v to proposal.approvals
        numApprovals = proposal.approvals.length()
        Q = 60% * n(ODAOMT)
        if numApprovals > Q then
            Set proposal.state = Executable
        end if
    end if
end procedure
```

"Executable". This step is outlined in procedure *approve* (Algo. 4).

- Step 3: After obtaining necessary approvals on a proposal, FLTP triggers its execution by sending an "execute" transaction to the MultiSigC with the unique $proposalID$. The MultiSigC validates the proposal's executability based on the proposal state ($proposal.state$) and MultiSigC state. If conditions are met, the MultiSigC executes the proposal within DAOFLC, updating its state. This process is outlined in procedure *execute* (Algo. 4). Following execution, the FLTP proposes the next "propose" transaction to continue DAO-FL operations in alignment with the FL process.

If a proposal lacks sufficient approvals due to inaccuracies in $tokenURI$ and $GMCID$, FLTP can create precise alternative proposals. To close inaccurate proposals, FLTP submits a "closeProposal" transaction with the relevant $proposalID$ as an argument, discarding the inaccurate proposal for future accurate ones. This process is outlined in the procedure

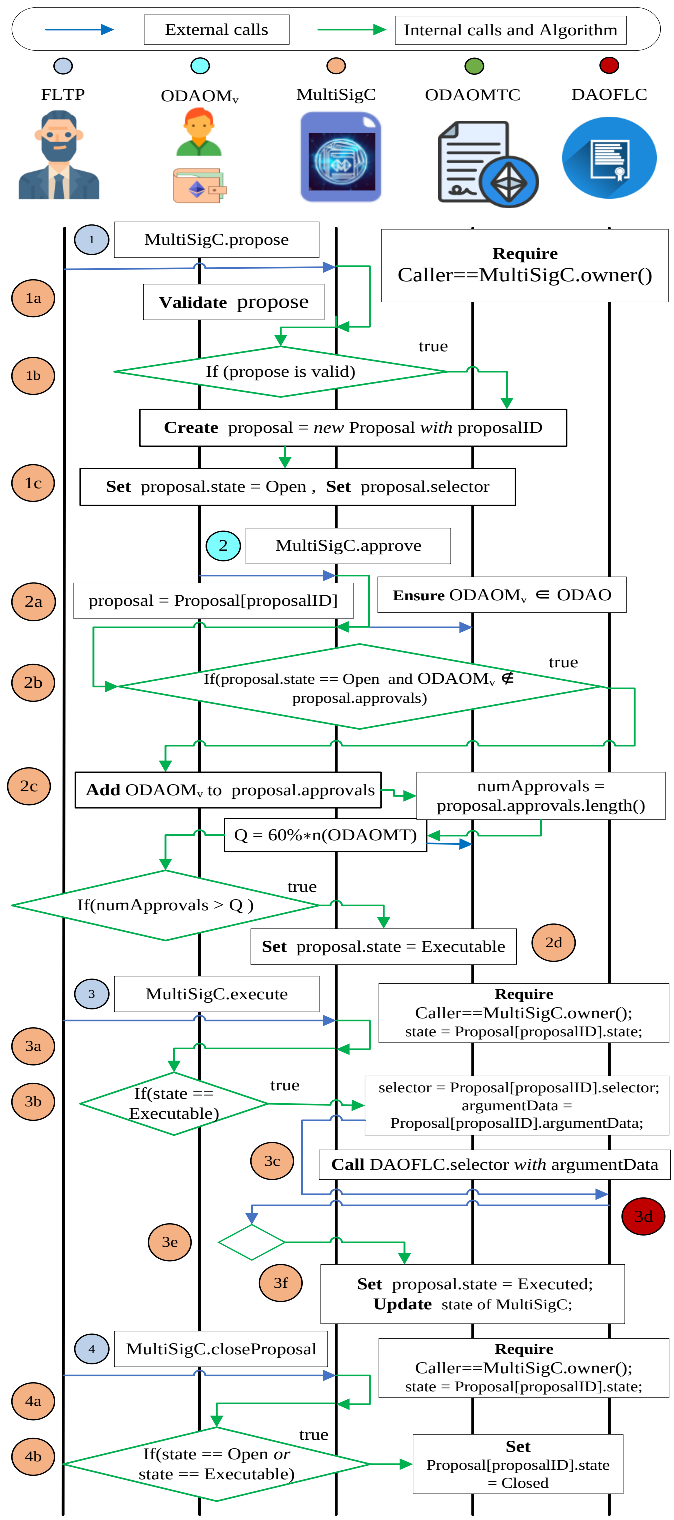

Fig. 4. Partially Decentralized Orchestration of FL process in DAOFLC through MultiSigC - Sequence diagram.

$closeProposal$ (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 concise outline of the sequential flow:

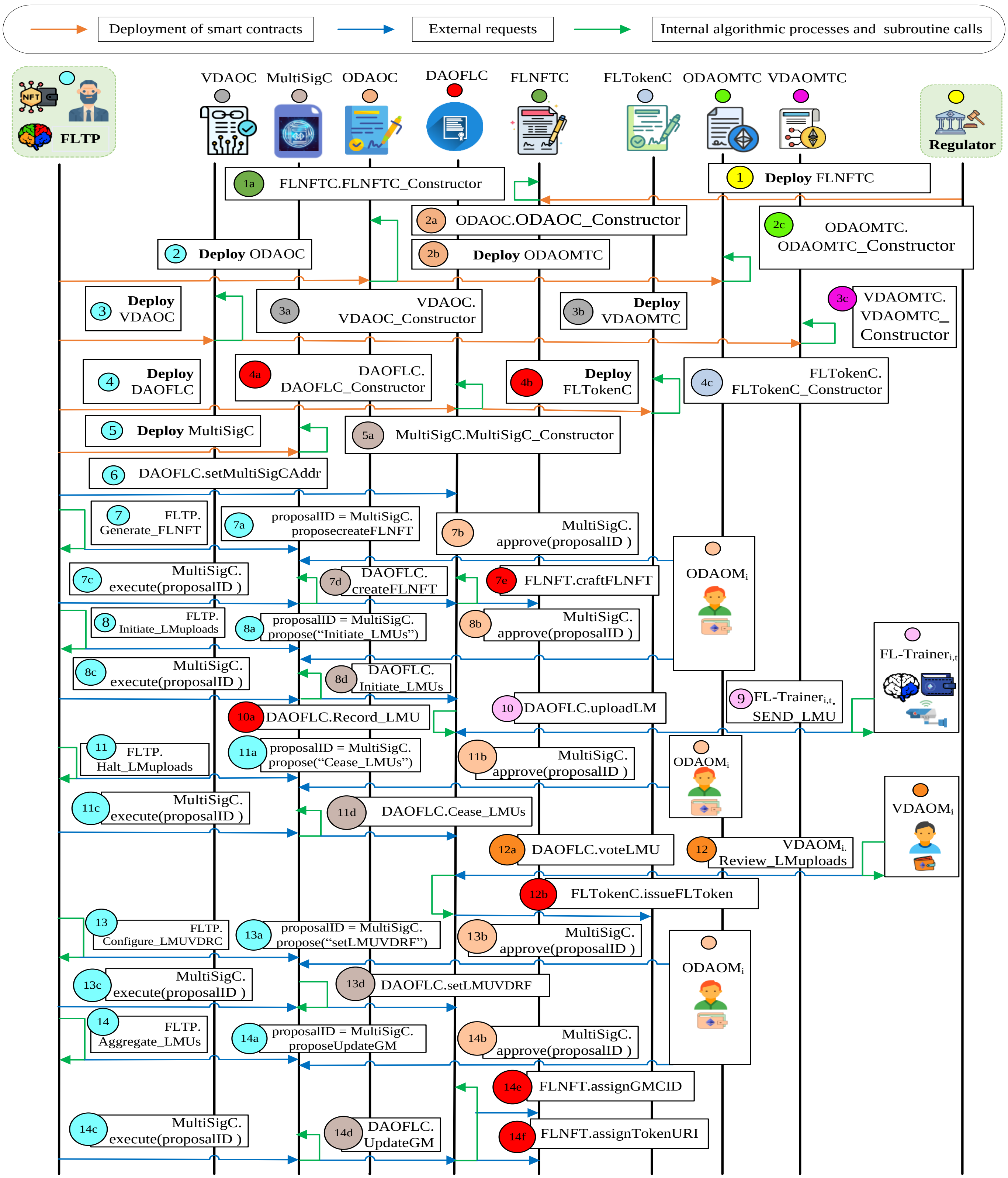

Fig. 5. DAO-FL: Simplified execution workflow.

- Step 1: The Regulator deploys the FLNFTC for the FL ecosystem. The deployment transaction includes three arguments: "Federated Learning NFT" as the name, "FLNFT" as the symbol, and a base_URI used in the TokenURI of FLNFTs. The ownership of FLNFTC is then transferred to the *Regulator*. The procedure $FLNFTC\_Constructor$ (Algo. 5) summarizes this step.
- Step 2: For the FL task, FLTP deploys the ODAOC, specifying two candidate ODAOMs ($ODAOM_i$) as arguments, along with a base_URI parameter for the TokenURI of ODAOMTs. The procedure

Algorithm 5 : FLNFTC

Owner: $Regulator$ **Deployer:** $Regulator$
Input: "Federated Learning NFT", "FLNFT", base_URI
procedure FLNFTC_Constructor(_name, _symbol, base_URI)
  Assign $FLNFTC.owner \leftarrow Regulator.address$
  Assign $FLNFTC.name \leftarrow \_name$
  Assign $FLNFTC.symbol \leftarrow \_symbol$
  Assign $FLNFTC.base\_URI \leftarrow base\_URI$
end procedure

Executor: DAOFLC
procedure craftFLNFT($GMCID$, $tokenURI$)
  $FLNFTID$ = **Mint** FLNFT transferred to $FLTP$
  Assign $FLNFT.tokenURI \leftarrow tokenURI$
  Assign $FLNFT.GMCID \leftarrow GMCID$
  Assign $FLNFT.OrchestratorAddress \leftarrow DAOFLC.address$
end procedure
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
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

procedure Generate_FLNFT
  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

procedure Initiate_LMuploads
  Call $MultiSigC.propose$ (selector, $t + 1$) ▷ selector for proposal "Initiate_LMUs"
end procedure

procedure Halt_LMuploads
  Call $MultiSigC.propose$ (selector, $t + 1$) ▷ selector for proposal "Cease_LMUs"
end procedure

procedure Configure_LMUVDRF
  Call $MultiSigC.propose$ (selector, $t + 1$) ▷ selector for proposal "setLMUVDRF"
end procedure

procedure Aggregate_LMUs
  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

Algorithm 7 : ODAOC

Owner: $FLTP$ **Deployer:** $FLTP$
procedure $ODAOC\_Constructor$(**address** $member1$, **address** $member2$, base_URI)
  Set $ODAOC.owner = FLTP.address$
  Deploy $ODAOMTC$ ("Orchestrator-DAOMT","ODAOMT", base_URI)
  Call $ODAOMTC.mint(FLTP)$
  Call $ODAOMTC.mint(member1)$
  Call $ODAOMTC.mint(member2)$
end procedure

Algorithm 8 : ODAOMTC

Owner: $ODAOC$ **Deployer:** $ODAOC$
Input: _name, _symbol, base_URI
procedure ODAOMTC_Constructor
  Set $ODAOMTC.owner = ODAOC.address$
  Set $ODAOMTC.name = \_name$
  Set $ODAOMTC.symbol = \_symbol$
  Set $ODAOMTC.base\_URI = base\_URI$
end procedure
Caller: ODAOC **Modifier:** onlyOwner()
procedure $mint$(**address** $recipent$)
  if $candidate \notin ODAO$ **then**
    Mint $ODAOMT$ for $recipent$
  end if
end procedure

$ODAOC\_Constructor$ (Algo. 7), is initiated for ODAOC deployment, transferring ODAOC's ownership to FLTP. ODAOC then deploys ODAOMTC with specified parameters (name, symbol, and base_URI of ODAOMTs), transferring its ownership to ODAOC. Subsequently, ODAOC mints ODAOMTs for FLTP and two specified members following the procedures $ODAOMTC\_Constructor$ and $mint$ (Algo. 8). Once ODAOC is deployed, the ODAOMs can perform membership enrollment and expulsion operations within ODAOC, as defined in Section IV-B and Section IV-C, respectively.

- Step 3: FLTP deploys the VDAOC using the procedure $VDAOC\_Constructor$ (Algo. 9), adding two entities as VDAO members ($VDAOM_i$), and taking ownership of VDAOC. A base URI is specified for the TokenURI of VDAOMTs during deployment. Following the procedures $VDAOMTC\_Constructor$ and $mint$ (Algo. 10), VDAOC deploys VDAOMTC with the provided arguments (name, symbol, and base URI of VDAOMTs), transferring VDAOMTC's ownership to VDAOC, and minting VDAOMTs for FLTP and the two specified members. Once VDAOC is deployed, VDAOMs can perform the membership enrollment and membership expulsion operations within VDAOC.
- Step 4: FLTP deploys DAOFLC with addresses of FLNFTC, ODAOC, and VDAOC as arguments, transferring DAOFLC's ownership. DAOFLC then deploys FLTokenC with a specific name and symbol for FLTokens, transferring FLTokenC's ownership to DAOFLC. This process is outlined in procedures $DAOFLC\_Constructor$ (Algo. 11) and $FLTokenC\_Constructor$ (Algo. 16).
- Step 5: The FLTP deploys the MultiSigC,

Algorithm 9 : VDAOC

Owner: $FLTP$ **Deployer:** $FLTP$
procedure VDAOC_Constructor(**address** $member1$, **address** $member2$, base_URI)
  Set $VDAOC.owner = FLTP.address$
  Deploy $VDAOMTC$ (“Validation-DAOMT”,“VDAOMT”, base_URI)
  Call $VDAOMTC.mint(FLTP)$
  Call $VDAOMTC.mint(member1)$
  Call $VDAOMTC.mint(member2)$
end procedure

Algorithm 10 : VDAOMTC

Owner: $VDAOC$ **Deployer:** $VDAOC$
Input: _name, _symbol, base_URI
procedure VDAOMTC_Constructor
  Set $VDAOMTC.owner = ODAOC.address$
  Set $VDAOMTC.name = \_name$
  Set $VDAOMTC.symbol = \_symbol$
  Set $VDAOMTC.base\_URI = base\_URI$
end procedure
Caller: VDAOC **Modifier:** onlyOwner()
procedure $mint$(**address** $recipent$)
  if $candidate \notin VDAO$ **then**
    Mint $VDAOMT$ for $recipent$
  end if
end procedure

transferring its ownership, as shown in procedure $MultiSigC\_Constructor$ (Algo. 13).

- Step 6: The FLTP submits the transaction “setMultiSigCAddr“ to DAOFLC with the address of MultiSigC as an argument. The procedure $setMultiSigCAddr$ (Algo. 11) summarizes this step. After this transaction, MultiSigC will be able to execute transactions in DAOFLC.
- Step 7: Following procedure $Generate\_FLNFT$ in (Algo. 6), the FLTP constructs the “preliminary GM parameters“ for the FL task and stores it on IPFS, which yields a CID referred to as $GMCID$. These parameters serve as GM_t for $t = 0$. Additionally, FLTP uploads relevant files, including instructions for FL tasks, LMUs, reward criteria, and any tailored information, to IPFS. All of these details, including the addresses of associated contracts, are encompassed within a JSON-encoded meta-data identified as $FLNFT\_Metadata_t$. The $FLNFT\_Metadata_t$ is uploaded to IPFS, resulting in a CID called $tokenURI$. Afterward, the FLTP initiates procedure $proposecreateFLNFT$ ($propse$) in Algo. 4 with arguments e.g. $tokenURI$ and $GMCID$. This initiates the multi-signature process as detailed in Section IV-E for proposal “createFLNFT”. During the execution of proposal “createFLNFT”, the DAOFLC mints the FLNFT on FLNFTC for FLTP, as illustrated in procedure $createFLNFT$ (Algo. 11) and procedure $craftFLNFT$ (Algo. 5). The corresponding properties including the $OrchestratorAddress$ of FLNFT are also set.
- Step 8: The FLTP triggers the procedure $Initiate\_LMuploads$ (Algo. 6) to commence the LMUs on the $DAOFLC$. FLTP initiates the procedure $propose$ (Algo. 4) with parameters like $selector$ and

Algorithm 11 : DAOFLC

Owner: $FLTP$ **Deployer:** $FLTP$
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()
procedure setMultiSigCAddr(MultiSigC.address)
  Set $DAOFLC.MultiSigCAddr = MultiSigC.address$
end procedure

Caller: $MultiSigC$ **Modifier:** onlyMultiSigC()
procedure createFLNFT($tokenURI$, $GMCID$)
  $FLNFTID$ = **call** $FLNFTC.craftFLNFT$ ($tokenURI$, $GMCID$)
  Set $DAOFLC.FLNFTID$ = $FLNFTID$
  Set $DAOFLC.GMCID$ = $GMCID$
end procedure
procedure Initiate_LMUs($t+1$)
  if $DAOFLC.LMUactiveF == false$ **then**
    Set $DAOFLC.LMUactiveF = true$
    Emit $DAOFLC.LMUsInitiated(t+1)$
  end if
end procedure
procedure Cease_LMUs(t+1)
  if $LMUactiveF == true$ **then**
    Set $LMUactiveF = false$ and $LMUC[t+1] = true$
    Emit $LMUsCeased(t+1)$
  end if
end procedure

Caller: $FLTrainer_{i,t+1}$
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

Input: $LMCID$, $LMURI$, $t+1$, $FLTrainer_{i,t+1}.address$
procedure Record_LMU
  LM = **Create** *new* LMUs[t+1][$FLTrainer_{i,t+1}.address$]
  Set $LM.status = Submitted$
  Set $LM.LMCID = LMCID$
  Set $LM.LMURI = LMURI$
  Set $LM.approvalvotes = LM.denyvotes = 0$
  Set $LM.voters$ = *empty* AddressSet
end procedure

$t+1$, where $selector$ is derived from the Keccak-256 hash of the “Initiate_LMUs” function signature in the DAOFLC. This starts the multi-signature process outlined in Section IV-E for the proposal “Initiate_LMUs”. During its execution, the procedure $Initiate\_LMUs$ (Algo. 11) checks the status of the $DAOFLC.LMUactiveF$ flag. A true value indicates that LMUs are accepted, while a false signifies LMUs closure. If $DAOFLC.LMUactiveF$ is false, it is updated to true and the $LMUsInitiated(t+1)$ event is emitted, indicating 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

Algorithm 12 : $DAOFLC$ - Continued

Caller: $VDAOM_i$ **Modifier:** onlyVDAOM
procedure voteLMU($FLTrainer_{i,t+1}.address$, $t+1$, V_i)
  Require:$VDAOM_i \notin LMUs[t+1]$ $[FLTrainer_{i,t+1}.address].voters$
  if $V_i == true$ **then**
    Add Approval vote for $VDAOM_i$
  else
    Add Deny vote for $VDAOM_i$
  end if
  Count $LM.approvalvotes$ and $LM.denialvotes$
  Q = 60% $* n(DAOMT)$
  if $LM.approvalvotes > Q$ **then**
    Call $FLTokenC.issueFLToken(FLTrainer_{i,t+1})$
    Set $LMU_{i,t+1}.status == Rewarded$
  else if $LM.denialvotes > Q$ **then**
    Set $LMU_{i,t+1}.status == Denied$
  end if
end procedure

Caller: $MultiSigC$ **Modifier:** onlyMultiSigC
procedure setLMUVDRF($t+1$)
  Set $LMUVDRF[t+1] = true$
end procedure
procedure UpdateGM($t+1$, $GMCID$, $tokenURI$)
  GMCIDsuccessF = **Call** $FLNFTC.assignGMCID($ $GMCID, FLNFTID)$
  TokenURIsuccessF = **Call** $FLNFTC.assignTokenURI($ $tokenURI, FLNFTID)$
  if GMCIDsuccessF **and** TokenURIsuccessF **then**
    Emit $GMupdated(t+1, GMCID, tokenURI)$
    Set $DAOFLC.tokenURI$ = $tokenURI$
    Set $DAOFLC.GMCID$ = $GMCID$
    Set $GIC[t+1] = true$
  end if
end procedure

Algorithm 13 : MultiSigC

Owner: $FLTP$ **Deployer:** $FLTP$
Input: $DAOFLC.address$, $ODAOC.address$
procedure MultiSigC_Constructor
  Set $MultiSigC.owner = FLTP.address$
end procedure

$SEND\_LMU$ (Algo. 14) 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 $D_{i,t+1}$, $FLTrainer_{i,t+1}$, they compute 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, resulting in the associated CID $LMCID$. Additionally, the JSON-encoded meta-data for $LM_{i,t+1}$ is generated and stored on IPFS, obtaining the CID $LMURI$. $FLTrainer_{i,t+1}$ submits its LMU to DAOFLC, using procedure $uploadLM$ (Algo. 11), with $LMCID$ and $LMURI$ as arguments. DAOFLC may impose a limit on the number of LMUs allowed for GI $t+1$.

- Step 10: The procedure $uploadLM$ (Algo. 11) is instigated by $FLTrainer_{i,t+1}$. DAOFLC.Authenticate_LMU function validates the $LMU_{i,t+1}$, potentially rejecting it if the LMUs limit is reached. If valid, $LMU_{i,t+1}$ is appended to the LMUs for GI $t+1$ and associated with the $FLTrainer_{i,t+1}$ via procedure $FLTPC.Record\_LMU$ (Algo. 11). LM properties, such as approval and deny votes, are set to 0, LM's voter list is set to empty and LM's status is marked as "Submitted".
- Step 11: The FLTP commences the procedure $Halt\_LMuploads$ (Algo. 6) to cease LMUs on the DAOFLC. This instigates the procedure $propose$ (Algo. 4) with arguments like $selector$ and $t+1$, where $selector$ represents the selector for the DAOFLC's "Cease_LMUs". This triggers the multi-signature process as detailed in Section IV-E for the "Cease_LMUs" proposal. The execution of this proposal activates the procedure $Cease\_LMUs$ (Algo. 11). If $LMUactiveF$ is true, it is changed to false, emitting the $LMUceased(t+1)$ event. The $LMUC$ flag is set as true, indicating the cessation of LMUs for GI $t+1$, and FL-Trainers halt LM uploads.
- Step 12: After LMUs are ceased for $t+1$, VDAOMs in VDAO concurrently initiate the procedure $Review\_LMuploads$ (Algo. 15). 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. 12) 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

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

and

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

respectively. The quorum Q is determined. If the $LM_{approvalvotes}$ exceed the Q, the procedure $FLTokenC.issueFLToken$ (Algo. 16) is utilized to issue an FL-Token for $FLTrainer_{i,t+1}$, and the LM status is set to "Rewarded". However, if the $LM_{denialvotes}$ exceed the Q, the LM status is set to "Denied".
- Step 13: The FLTP initiates the procedure $Configure\_LMUVDRC$ (Algo. 6). Using the $selector$ of the "setLMUVDRF" function within the DAOFLC and $t+1$ as arguments, the FLTP triggers the multi-signature process as outlined in Section IV-E for the "setLMUVDRF" proposal by invoking procedure $propose$ (Algo. 4). As part of executing this proposal, the procedure $setLMUVDRF$ (Algo. 12) is activated, setting the flag $LMUVDRF(t+1)$ for GI $t+1$ to

Algorithm 14 : FL-Trainer $FLTrainer_{i,t+1}$

```
procedure SEND_LMU
    Get DAOFLC.GMCID
    Download GM_t ← IPFS using DAOFLC.GMCID
    Generate LM_{i,t+1} using [6]
    LMCID = Store LM_{i,t+1} on IPFS
    Create LMURI for LM_{i,t+1}
    LMURI = Store LMURI on IPFS
    Call DAOFLC.uploadLM(LMCID, LMURI, t + 1)
end procedure
```

Algorithm 15 : VDAO member $VDAOM_i$

```
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 16 : FLTokenC

```
    Owner: DAOFLC        Deployer: DAOFLC
procedure FLTokenC_Constructor(_name, _symbol)
    Set FLTokenC.owner = DAOFLC.address
    Set FLTokenC.name = _name
    Set FLTokenC.symbol = _symbol
end procedure
procedure issueFLToken(FLTrainer_{i,t+1}.address)
    Mint 1 * 10^18 FLToken for FLTrainer_{i,t+1}
end procedure
```

signal the completion of LMUs' verification, denial, or reward process.

- 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]:

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

where $\boldsymbol{w}_{t+1}^i$ is local parameter, $\boldsymbol{w}_{t+1}$ is global parameter, $n_i = |\mathcal{D}_i|$, and $n = \left|\bigcup \mathcal{D}_i\right|$ and stores it on IPFS, yielding in CID $GMCID$. The updated meta-data, encoded in JSON format, denoted as $FLNFT\_Metadata_{t+1}$, is created and stored on IPFS, resulting in CID $tokenURI$. The FLTP then proposes the "UpdateGM" using the procedure $proposeUpdateGM$ ($propose$) in Algo. 4 with arguments such as $t + 1$, $GMCID$, and $tokenURI$. This triggers the multi-signature process outlined in Section IV-E for the proposal. During this process, ODAOMs aggregate $\hat{LMU}_{t+1}$ following predefined guidelines and approve the proposal to certify its authenticity and accuracy. During the execution of the proposal, the procedure $UpdateGM$ (Algo. 12) is called. This procedure sets the $GMCID$ and $tokenURI$ of the FLNFT by invoking the procedures $FLNFTC.assignGMCID$ and $FLNFTC.assignTokenURI$ (Algo. 5) respectively [17]. Only the registered $OrchestratorAddress$ can execute these procedures. The $FLNFTC.assignGMCID$ verifies the submitted $GMCID$ using the FLNFTC.Verify_GMCID function, ensuring unique GMCIDs across all FL-NFTs. Similarly, the $FLNFTC.assignTokenURI$ verifies the submitted $tokenURI$ using the FLNFTC.Verify_TokenURI function, ensuring unique tokenURIs for all FL-NFTs. The DAOFLC emits the event $DAOFLC.GMupdated$, and the $GIC[t + 1]$ is flagged to indicate the completion of GI $t + 1$.

Algorithm 17 : FL-NFT's transfer

```
    Caller: FLTP
procedure FLNFT_TRANSFER(new_owner)
    Require: new_owner != FLTP.address
    Call FLNFTC.transferFrom(FLTP.address,
new_owner, FLNFTID )
    Call ODAOC.transferOwnership(new_owner )
    Call VDAOC.transferOwnership(new_owner )
    Call MultiSigC.transferOwnership(new_owner )
    Call DAOFLC.transferOwnership(new_owner )
end procedure
```

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

G. Commercializing GM and Transferring ownership

The GM is tokenized to manage FL processes efficiently and enable potential commercialization through platforms like OpenSea. The trading involves transferring the FLNFT of GM to the buyer. However, in DAO-FL, the FLTP, who owns the FLNFT, also owns contracts like DAOFLC, MultiSigC, ODAOC, and VDAOC. To transfer GM's ownership to a new proprietor, the FLTP initiates the procedure $FLNFT\_Transfer$ (Algo. 17). This involves transferring the FLNFT as well as ownership of DAOFLC, MultiSigC, ODAOC, and VDAOC to the new owner.

V. IMPLEMENTATION, DEPLOYMENT, AND EVALUATION

In this section, we present the implementation, deployment, and evaluation aspects of the DAO-FL framework.

A. Implementation and Deployment

The smart contracts for the DAO-FL framework were developed using the Solidity programming language [24]. To visualize the inheritance hierarchy of these contracts, we utilized the Surya tool [25]. To enable membership in ODAO and VDAO, we required a token standard known as NTT, such as EIP-4671 [26]. However, as NTT tokens are still in the early stages of development and might not meet our specific requirements, we created a custom smart contract called "DAOMTC" to implement DAOMTs. The inheritance graph of DAOMTC, illustrated in Fig. 6, demonstrates that DAOMTC is inherited from customized OpenZeppelin [27] "Ownable" [28] and "ERC165" contracts. Additionally, DAOMTC implements the

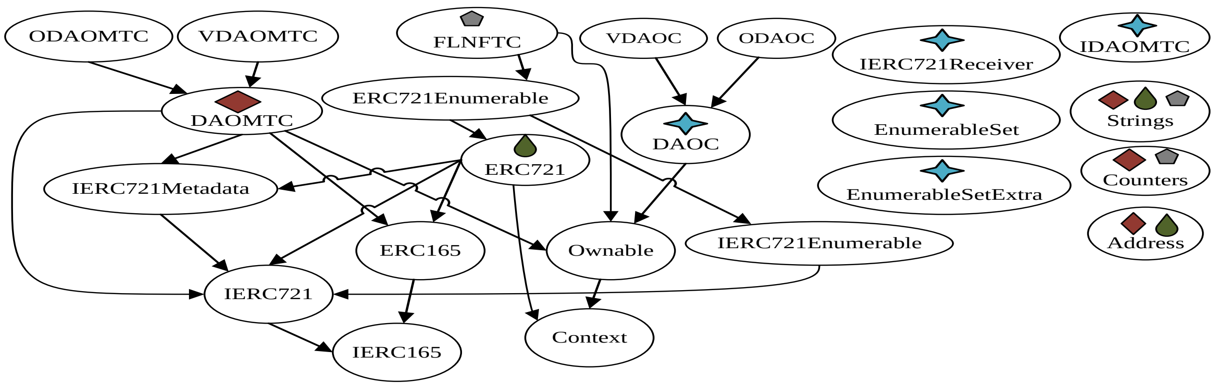

Fig. 6. Inheritance graph of the DAOC, DAOMTC, ODAOC, VDAOC, ODAOMTC, VDAOMTC, and FLNFTC.

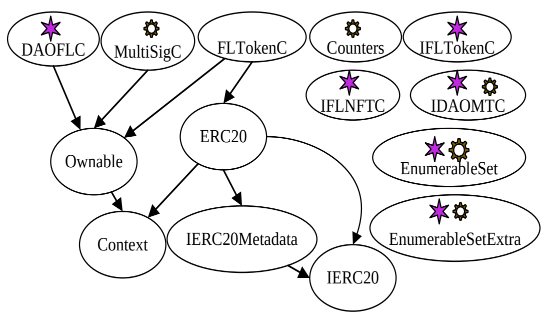

Fig. 7. Inheritance graph of the DAOFLC, MultiSigC, and FLTokenC.

IERC721Metadata interface. Since DOAMTs are NTT, certain functions of the IERC721 interface are not applicable but are included for compatibility with NFT-related platforms like OpenSea. For efficient membership management in ODAO and VDAO, we have introduced a specialized smart contract named DAOC. By implementing generalized procedures for adding or removing members in a DAO, DAOC serves the purpose of both ODAO and VDAO. Inheritance-wise, DAOC extends a customized "Ownable" contract [28], which itself inherits from the "Context" contract [28]. Appendix B includes the class diagram for DAOMTC and DAOC.

ODAO and VDAO are two distinct DAOs implemented in ODAOC and VDAOC, respectively. These DAOs utilize ODAOMTs and VDAOMTs as their respective membership tokens. ODAOMTs and VDAOMTs are implemented in ODAOMTC and VDAOMTC respectively. The inheritance graph in Fig. 6, reveals that ODAOC and VDAOC inherit from DAOC, while ODAOMTC and VDAOMTC inherit from DAOMTC. The detailed representation of the class diagrams for ODAOC, ODAOMTC, VDAOC, and VDAOMTC is provided in Appendix C.

FLNFTC inherits functionalities from two sources: the ERC721Enumerable standard [29] and the "Ownable" contract [28]. Fig. 6 depicts the inheritance graph of FLNFTC. FLTokenC is derived from the "Ownable" contract [28] and the OpenZeppelin [27] ERC-20 implementation [30]. Both DAOFLC and MultiSigC inherit from the "Ownable" contract [28]. Fig. 7 illustrates the inheritance graph for DAOFLC, MultiSigC, and FLTokenC. For a detailed class diagram for DAOFLC, FLTokenC, FLNFTC, and MultiSigC, please refer to Appendix C.

The smart contracts underwent compilation using the Hardhat [31]. The smart contracts were deployed on the Sepolia testnet [32]. To ensure transparency, the deployed smart contracts on the Sepolia network were verified using the ETHERSCAN_API_KEY [33]. The gas utilized, gas price, and transaction fee (in ethers) for deploying smart contracts are illustrated in Fig. 8. It should be noted that the gas used for ODAOMTC, VDAOMTC, and FLTokenC is encompassed within the gas used for ODOAC, VDOAC, and DAOFLC, respectively. For FLNFTC, the gas price was approximately 0.15 Gwei, which was comparatively high, possibly due to network congestion during its deployment. As a result, the elevated gas price led to a transaction fee of 0.00032 ETH. Consequently, the gas price and transaction fee for FLNFTC are not depicted in Fig. 8.

The Etherscan links of key entities (Regulator, FLTP, and $FLTrainer_{1,1}$) and smart contracts deployed on the Sepolia network are presented in Table III. By examining these addresses on Etherscan Explorer, users can gain access to comprehensive information including event logs, internal and external transaction logs, and verified contract codes [17]. Given the broader focus of our paper on establishing a decentralized ecosystem for IV and OV of FL process through multi-signature wallets and DAOs, we utilized the MNIST, Fashion-MNIST, and UNB ISCX VPN-NonVPN network traffic [34] dataset for training the local and global models. Consequently, we will omit specific details related to model configuration, accuracy information, and data allocation in this context. Due to space constraints, some repetitive transactions required to reach quorum have been omitted in some onward figures for brevity.

As the procedures for member enrollment and expulsion are the same for ODAOC and VDAOC, we present the implementation results for ODAOC. Fig. 9 illustrates the transaction list for a "Join Proposal" (JP), including the "proposeJoin" transaction initiated by $ODAOM_p$ and the "voteJoin" transactions by ODAOMs. It also captures the relevant events emitted by ODAOC, such as JPsubmitted, JPdenialVote, and JPapprovalVote. Additionally, Fig. 10 showcases the minting of ODAOMT upon reaching the quorum, accompanied by the events "JPapproved" emitted by ODAOC to indicate JP approval and the "Transfer" event indicating the transfer

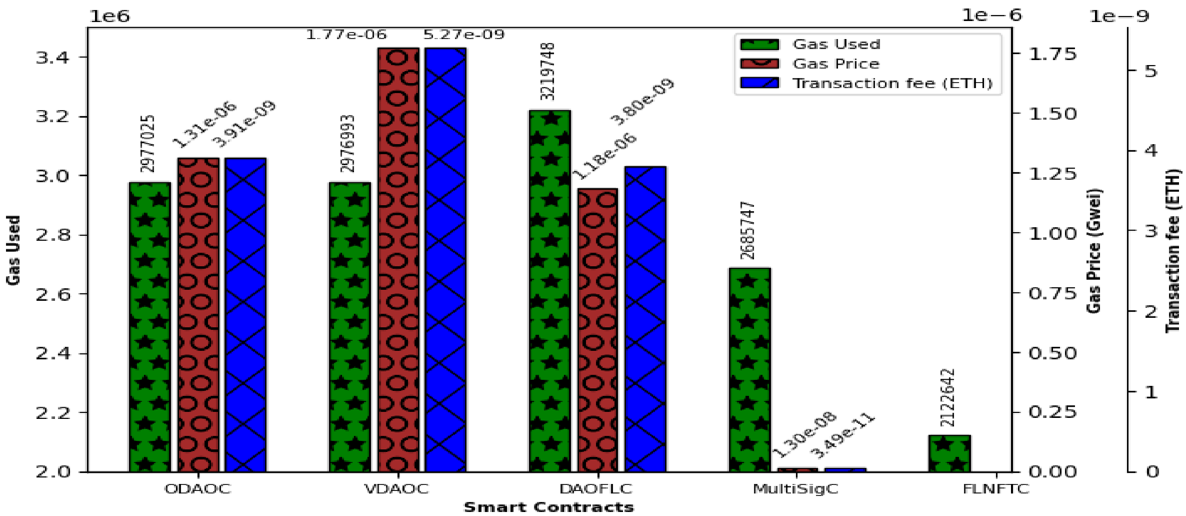

Fig. 8. Gas Used, Gas Price, and transaction fee (in ETH) for the deployment of smart contracts.

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

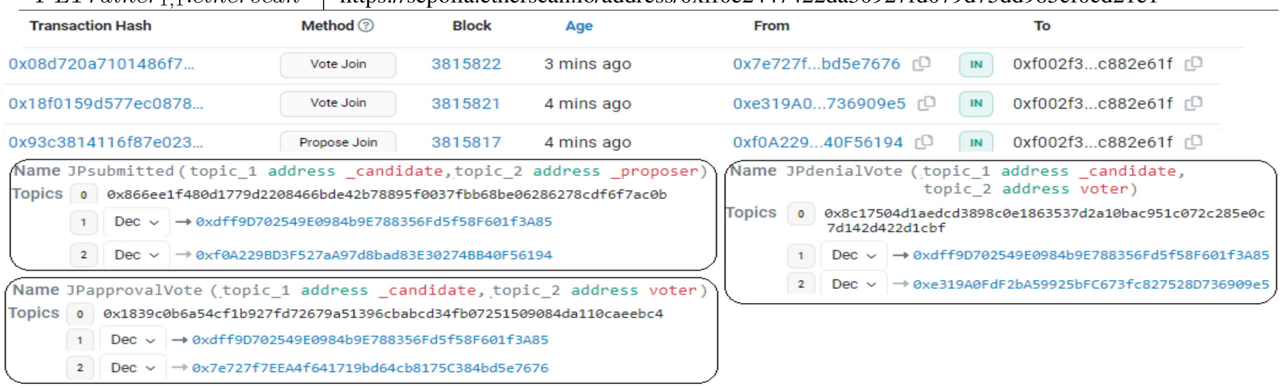

Fig. 9. Transaction sequence ($DAOC.proposeJoin$ and $DAOC.voteJoin$) and emitted events for a "Join Proposal" on ODAOC (https://sepolia.ethersca n.io/address/0xf002f304Cb1C34b40d59347472f2f68Fc882e61f), [Block 3815817-3815822].

Fig. 10. 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).

of ODAOMT to the *candidate*, emitted by ODAOMTC. Similarly, Fig. 11 presents the transaction list for a "Kick Proposal" (KP), which includes the "proposeKick" transaction initiated by $ODAOM_p$ and the "voteKick" transactions by ODAOMs. The associated events emitted by ODAOC, such as KPsubmitted, KPdenialVote, and KPapprovalVote, are also captured. Furthermore, Fig. 12 showcases the burning of ODAOMT upon reaching the quorum, along with the events "KPapproved" emitted by ODAOC to indicate KP approval and the "Transfer" event signifying the burning of ODAOMT

Fig. 11. Transaction sequence ($DAOC.proposeKick$ and $DAOC.voteKick$) and emitted events for a "Kick Proposal" on ODAOC (https://sepolia.etherscan.io/address/0xf002f304Cb1C34b40d59347472f2f68Fc882e61f), [Block 3815823-3815828].

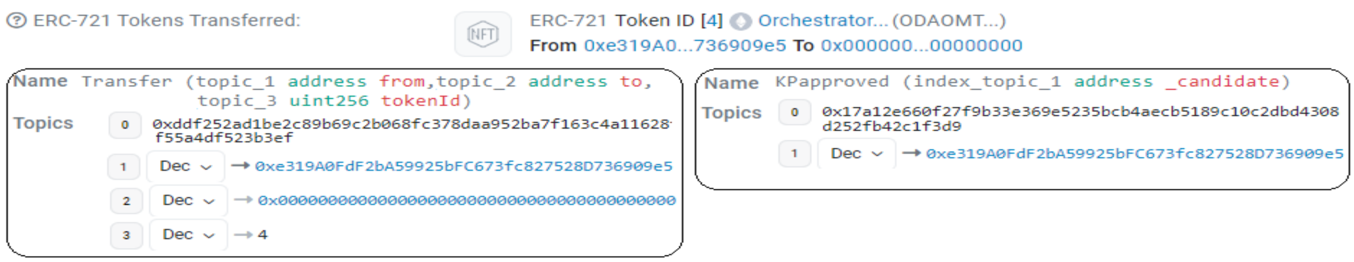

Fig. 12. Burning of ODAOMT after reaching the quorum of approval votes for "Kick Proposal" and corresponding events emitted on ODAOC (https://sepolia.etherscan.io/tx/0x7de873fc9bdfb1fca45ad560430eff5ee4778e821fd1e8d981c12a6f1c099da3).

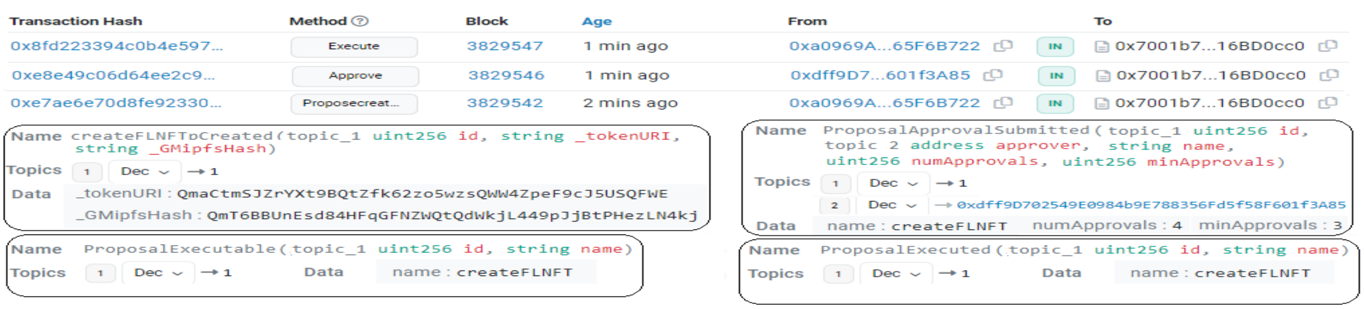

Fig. 13. Transaction sequence for the creation and execution of the "createFLNFT" proposal on MultiSigC, along with emitted events (https://sepolia.etherscan.io/address/0x7001b7f257EEDF4b970577c63095909916BD0cc0), Block [3829542-3829547].

owned by the $candidate$, emitted by ODAOMTC.

Onwards in this section, we present the implementation of the DAO-FL framework, following the steps outlined in Section IV-F. Fig. 13 depicts the creation of a "createFLNFT" proposal by FLTP using the procedure $FLTP.Generate\_FLNFT$ through the transaction "proposecreateFLNFT" on MultiSigC. It also includes one of the "approve" transactions by ODAOMs and the subsequent execution of the "createFLNFT" proposal by FLTP upon reaching quorum. The corresponding events emitted by MultiSigC, such as createFLNFTpCreated, ProposalApprovalSubmitted, ProposalExecutable, and ProposalExecuted, are also shown. Fig. 14 demonstrates the minting of FLNFT following the execution of the "createFLNFT" proposal. The events emitted by FLNFTC, including OrchestratorAddressSet, GMCIDset, and TokenURIset, are displayed. Additionally, the event FLNFTcreated emitted by DAOFLC is depicted. Fig. 15 illustrates the creation and execution of the "Initiate_LMUs" proposal by FLTP, following its approval by ODAOMs. The figure also includes the emitted events, such as ProposalCreated and ProposalExecuted by MultiSigC, and LMUsInitiated by DAOFLC. After listening to the LMUsInitiated event, $FLTrainers_{t+1}$ uploads LMs through the "uploadLM" transaction on DAOFLC, as depicted in Fig. 16. The event "LMuploaded" emitted by DAOFLC during a transaction is also shown.

The illustration of the creation and execution of the "Cease_LMUs" proposal will be omitted. However, after its execution, VDAOMs engage in the crucial task of IV for the FL process. This is achieved through the initiation of "voteLMU" transactions, as illustrated in Fig. 17. The events LMUvoted, LMURewarded, and LMUdenied are emitted by DAOFLC which signifies the validation process of LMUs. Furthermore, the successful validation results in the minting of FLTokens, 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 the execution of proposal "setLMUVDRF", FLTP submits proposal "UpdateGM" to MultiSigC as shown in Fig 18 where event UpdateGMpCreated is emitted. The proposal goes through the approval process by ODAOMs as DOV of the FL process and is

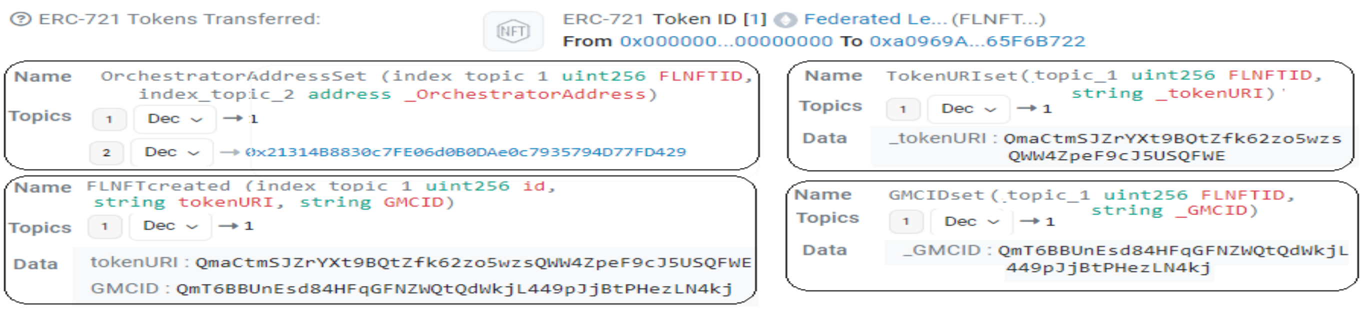

Fig. 14. Minting of FLNFT and emitted events during the execution of the "createFLNFT" proposal (https://sepolia.etherscan.io/tx/0x93e76ce42d9b76f6b4ede511e262e7ac9d77e5079f2cd0171e8e2e554d231a7a).

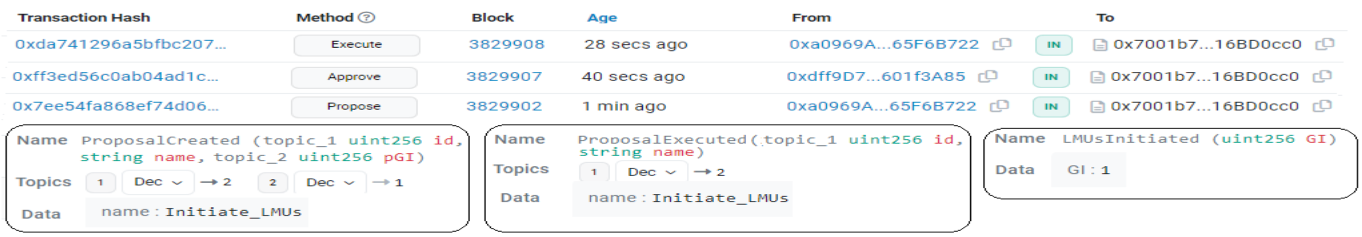

Fig. 15. Execution of the "Initiate_LMUs" proposal by FLTP, and emitted events by MultiSigC and DAOFLC (https://sepolia.etherscan.io/address/0x7001b7f257EEDF4b970577c63095909916BD0cc0), Block [3829902-3829908].

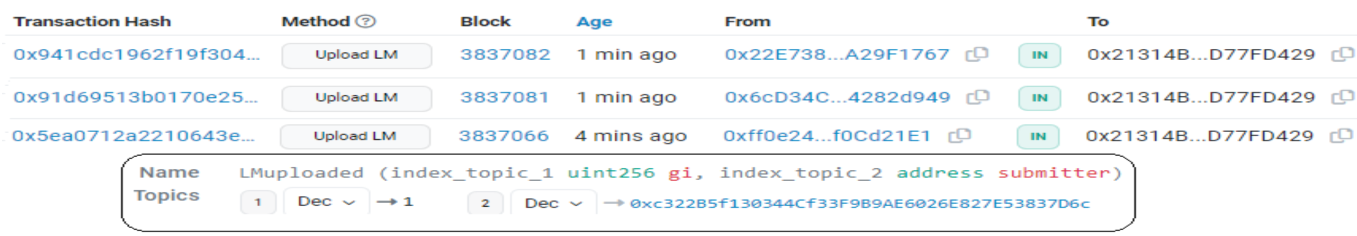

Fig. 16. Uploading of LM on DAOFLC by $FLTrainers_{t+1}$ (https://sepolia.etherscan.io/address/0x21314B8830c7FE06d0B0DAe0c7935794D77FD429) and event emitted, Block [3837066-3837082].

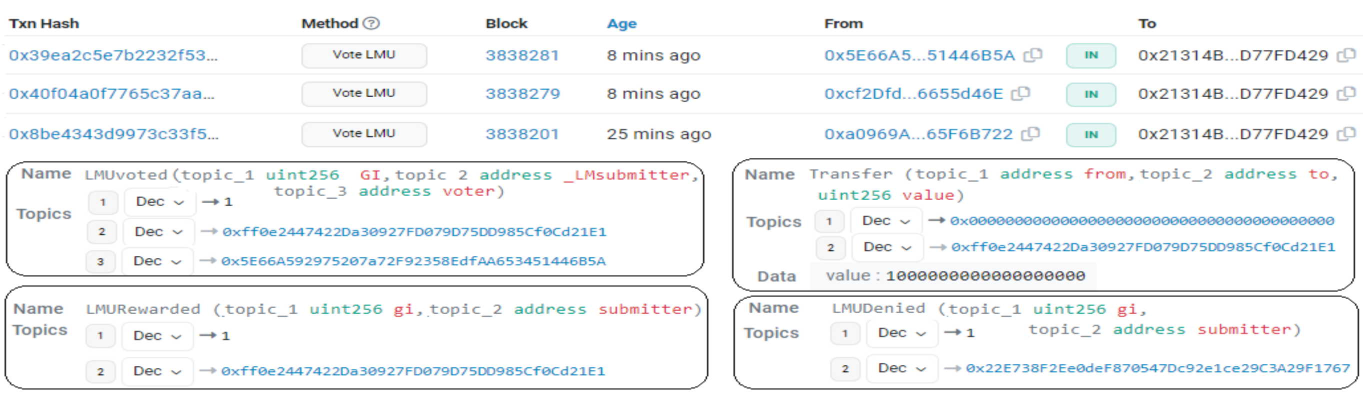

Fig. 17. Decentralized input verification of LMUs by VDAOMs for the FL process, minting of FLToken and other events emitted (https://sepolia.etherscan.io/address/0x21314B8830c7FE06d0B0DAe0c7935794D77FD429), Block [3838201-3838281].

finally executed. The events emitted are ProposalExecuted by MultiSigC, GMupdated by DOAFLC, and GMCIDset and TokenURIset by FLNFTC which shows that FLNFT has been updated.

B. Evaluation on Threat Models

In the context of information flows, vulnerabilities can arise at the input or output stages. Input vulnerabilities involve discrepancies between submitted inputs and prescribed policies. For the FL process, input attacks could manifest as submitting inaccurate or malicious LMs, potentially compromising the FL server to accept it for potential incorporation into upcoming GM, thereby jeopardizing GM's accuracy. Output vulnerabilities pertain to non-compliance of the produced outputs with information flow policies or post-production tampering. In the FL process, this output attack translates to scenarios like aggregation attacks or GM tampering after aggregation. Aggregation attacks occur when LMs are aggregated incorrectly to a GM. Post-production GM tampering occurs when the produced GM is replaced by a malicious one.

Fig. 19 depicts test accuracy trends of DAO-FL and centralized-FL subject to input, output, and input & output attacks on MNIST and Fashion-MNIST datasets for image classification and UNB ISCX VPN-NonVPN network traffic

Fig. 18. Creation and execution of proposal "UpdateGM" after DOV by ODAOMs (https://sepolia.etherscan.io/address/0x7001b7f257EEDF4b970577c6309 5909916BD0cc0) and events emitted, Block [3843770-3843775].

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

dataset [34] for network traffic classification [35] ($E = 10$ local epochs, $N = 10$ FL-Trainers per global epoch). Fig. 19(a, c, d, f, g, i) underscore DAO-FL's robustness to input attacks, as it rejects malicious LMs through DIV via VDAO, maintaining GM accuracy. DAO-FL closely matches the accuracy of attack-free-FL, particularly nearing convergence. The slight accuracy drop in DAO-FL (upon input attack) versus attack-free-FL results from the diversity of accurate LMs in attack-free-FL, while in DAO-FL, global parameters are biased towards approved LMs. In contrast, centralized-FL reliant on a single manipulable server, loses accuracy under input attacks. Fig. 19(b, c, e, f, h, i) show DAO-FL strictly maintaining accuracy under output attack. This resilience stems from the ODAO's vigilance through DOV by rejecting malicious "UpdateGM" proposals. The ODAO enforces the FLTP for alternative accurate "UpdateGM" proposals. Conversely, centralized FL, prone to tampering or aggregation attacks, experiences accuracy deterioration. These illustrations show that DAO-FL outperforms in countering input and output attacks.

Both input and output attacks in centralized FL lead to a decline in accuracy. After an attack, the GM may or may not be able to recover its original accuracy. These attacks compromise accuracy, introduce bias, or halt the FL process due to learning failures such as vanishing or exploding gradients. The learning failures are evident for centralized-FL in Fig. 19(c,f) at epoch=10 and Fig. 19(i) at epoch=250 onwards. Preventing these attacks is pivotal for the success of the FL process.

C. *Qualitative Evaluation and Discussion*

Our proposed framework provides a secure management solution for FL process. The involvement of multiple stakeholders, including regulators, FLTP, ODAO, and VDAO, facilitates decentralized governance and decision-making. This enables a more democratic and diverse approach to managing the FL process. DAO-FL framework utilizes smart contracts ODAOC and VDAOC to manage membership in ODAO and VDAO respectively. It leverages minting and burning of membership tokens for enrollment and expulsion procedures. These membership operations are decentralized relying on voting mechanisms to execute "Join Proposals" and "Kick Proposals".

DAO-FL is compatible with any underlying FL algorithm, just the validation of LMs and GM will be through DIV and DOV according to the prescribed security protocol of the FL algorithm. DAO-FL incorporates IV through the validation of LMUs by VDAOMs. This process enhances the trustworthiness of the FL process by allowing participants to verify the quality and integrity of the submitted LMs. The level of decentralization in ODAO is directly correlated with the total supply of ODAOMTC. As the total supply of ODAOMTC increases, the decentralization in OV of FL process also increases. Similarly, the decentralization in VDAO is directly tied to the total supply of VDAOMTC. An elevated total supply of VDAOMTC fosters increased decentralization in the IV of FL process. In scenarios prioritizing high decentralization, especially in prominent FL setups, the trade-off of increased time and high cumulative transaction fees to reach the quorum becomes acceptable as the ODAOMTC or VDAOMTC supply increases.

In DAO-FL, the ODAO only approves proposals in a decentralized fashion, The actual execution of these proposals remains under the responsibility of FLTP, resulting in a partially decentralized orchestration of the FL process. To attain full decentralization orchestration, a potential solution involves substituting FLTP with an the Executer-DAO, coupled with an appropriate multi-signature contract. This arrangement can facilitate the decentralized execution of approved proposals, thereby achieving a fully decentralized orchestration paradigm.

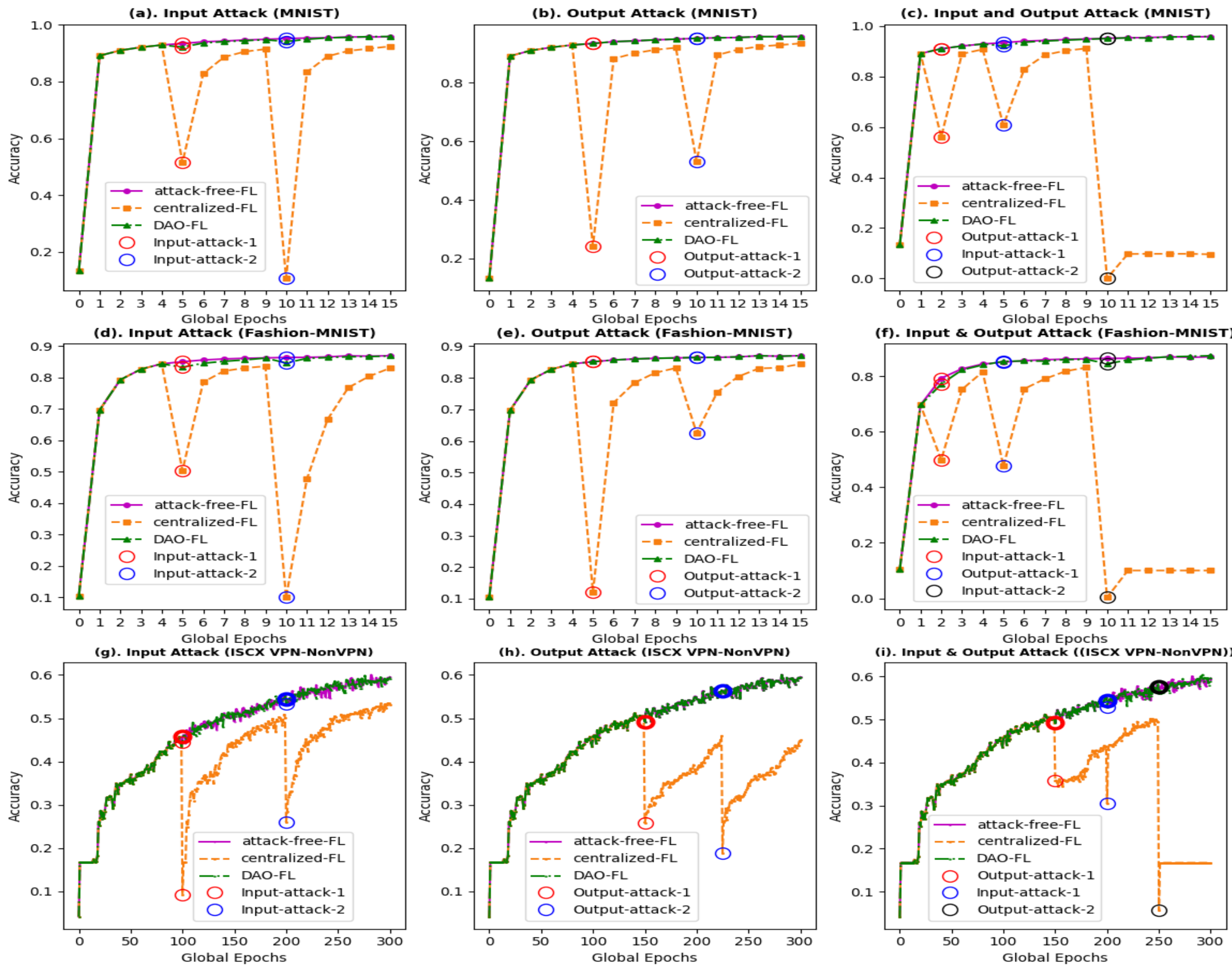

Fig. 19. Threat Evaluation of Input, Output, and Input & Output Attacks on DAO-FL, and centralized-FL (N=10, E=10).

FL inherently safeguards raw data access, with the GM typically considered secure against sophisticated data leaks. However, LMs remain vulnerable to data leaks through inference attacks. In DAO-FL, authentication via VDAOMT restricts access to LMs to authorized VDAO members through smart contract interfaces. While the proposed DAO-FL framework operates on a public blockchain, still posing a risk of access by malicious actors to LMs, integrating privacy-preserving techniques such as differential privacy at the LM level can bolster data security and mitigate the potential for data leaks.

The innovative principles and technologies embedded in DAO-FL offer a versatile framework that extends beyond its original context. Beyond FL, DAOMTs can be universally utilized as proof of membership in diverse DAOs. The proposed decentralized enrollment and expulsion schemes hold relevance across various DAO implementations. The versatility of smart contracts like MultiSigC and DAOFLC is evident, as with thoughtful adaptation of requirements and nomenclature of proposals to be executed, they can be used to enable partially decentralized orchestration for a wide spectrum of information flows. Additionally, the efficacy of the proposed quorum-based DIV and DOV mechanisms is not confined to the realm of FL alone. These mechanisms can be adapted to suit the specific needs of diverse information flows that necessitate decentralized decision-making, ensuring their applicability across a broad array of information flows.

D. Applicability, Limitations, and Future directions

With its decentralized governance and validation mechanisms, DAO-FL is particularly well-suited for industries that prioritize the integrity of the GM and the FL process. Sectors such as healthcare and finance, where privacy, transparency, and security are critical, can benefit greatly from DAO-FL's decentralized approach. By instilling trust in FL processes, DAO-FL ensures compliance with regulations and safeguards AI systems. Moreover, in industries like supply chain management and logistics, DAO-FL enables seamless collaboration among stakeholders while preserving GM integrity. Although DAO-FL, in its current proposed form, prioritize AI model security over rapid learning times and low costs, it offers significant advantages in sectors where data integrity is paramount. Furthermore, in industries susceptible to cyber attacks, such as critical infrastructure or defense, DAO-FL's decentralized validation mechanisms can effectively mitigate risks and enhance system resilience. This makes it a valuable solution for safeguarding intellectual property and ensuring the reliability of machine-learning models in high-risk environments.

Despite its potential benefits, DAO-FL faces significant limitations that impede its widespread adoption, particularly within real-time and time-sensitive applications. Industries such as high-frequency trading or autonomous vehicles, which demand low-latency decision-making, may find DAO-FL less applicable due to inherent complexities and higher associated transaction costs stemming from its decentralized nature. Notably, challenges such as non-deterministic response times, reliance on blockchain network constraints, intricate validation processes, and data transfer overhead, collectively pose obstacles to its suitability for swift decision-making scenarios. Furthermore, the decentralized nature of DAO-FL introduces variability in response times, rendering it unsuitable for real-time FL applications, thus underscoring the necessity for further research and optimization endeavors.

The cost analysis of DAO-FL is intricate due to the multitude of factors impacting transaction costs on a public blockchain. These expenditures are contingent upon variables such as gas fee, gas price, and network congestion, which lack determinism. The quantity of transactions necessitated to attain a quorum also relies on the existing supply of ODOMTs and VDAOMTs. Moreover, in times of heightened network congestion, gas prices have a propensity to escalate. Table IV lists the average transaction cost for FL-Incentivizer (centralized) and DAO-FL (decentralized) for IV and OV of an LM and a GM respectively on public (Sepolia) as well as private blockchain. The transaction cost for DIV and DOV is higher than centralized IV and OV on a public blockchain. Nevertheless, in the context of a private blockchain, transaction costs can be mitigated to zero. It is crucial to acknowledge that although private blockchains typically entail no transaction costs, there exists an initial setup cost linked to establishing the network infrastructure. Moreover, the transformation of GM into FLNFT for commercial purposes may not be viable on a private blockchain. The elevated transaction costs correlated with DAO-driven solutions on public blockchains frequently pose a common challenge attributable to on-chain voting mechanisms.

To mitigate the constraints of DAO-FL and enhance its viability across industries, various strategic avenues can be explored. Implementing off-chain voting mechanisms, exemplified by platforms like Snapshot and Aragon, can effectively reduce on-chain transaction costs associated with DIV and DOV. Additionally, techniques such as gas optimization and the adoption of layer-2 scaling solutions like state channels or Plasma hold promise in optimizing transaction fees and alleviating congestion on the primary blockchain network. Furthermore, concerted efforts to streamline validation processes, minimize data transfer overhead, and refine consensus mechanisms can collectively amplify the efficiency and responsiveness of DAO-FL, particularly in real-time applications. By proactively addressing these challenges and embracing innovative solutions, DAO-FL can transcend its limitations, thereby unlocking its full potential across diverse industry sectors.

E. Case Studies

These case studies or usage scenarios illustrate how DAO-FL can effectively prevent and respond to input and output attacks in FL in various real-world scenarios:

1) Inventory and Logistics Operations in Supply Chain Management: In the dynamic landscape of Supply Chain Management (SCM), DAO-FL emerges as a transformative solution by integrating DIV and DOV mechanisms to fortify FL processes. Imagine a scenario where multiple stakeholders in a global supply chain network collaborate in a FL setup to optimize inventory management and streamline logistics operations. By leveraging DAO-FL, these stakeholders securely share LMs for collaborative model training, ensuring the authenticity and integrity of inputs through decentralized validation. In this setting, malicious actors attempting to inject incorrect LMs or manipulate the GM face formidable barriers, as DAO-FL's robust verification protocols detect and respond to potential attacks swiftly, safeguarding the accuracy of SCM predictions and preserving the integrity of decision-making processes, thereby enhancing operational efficiency and resilience in the supply chain ecosystem.

2) Fraud Detection in Financial Institutions: In financial institutions, FL systems play a crucial role in detecting fraud while protecting customer confidentiality. To bolster fraud detection while adhering to privacy regulations, banks can collaborate to perform FL under regulatory oversight, such as the state bank, employing DAO-FL for collaborative fraud detection. At the end of a specified period (e.g., day, week, or month), participating banks train their LMs on the latest transaction data and securely share these models as per the DAO-FL guidelines. The GM is then generated based on the aggregated LMs. However, Malicious actors may submit incorrect LMs or perform output attacks on GM to destabilize the fraud detection system. By leveraging decentralized mechanisms such as DIV and DOV for verifying the reliability and transparency of LMs and GM updates, DAO-FL strengthens fraud prevention measures. This proactive approach aids in effectively detecting and responding to fraudulent activities, thereby safeguarding customer interests and upholding the integrity of financial transactions.

VI. Conclusion

This article proposed the DAO-FL framework, a groundbreaking approach to decentralized autonomous organizations for enhancing FL processes. By incorporating decentralized input verification and output verification mechanisms, DAO-FL ensures the integrity and security of the FL ecosystem. The utilization of DAO Membership Tokens (DAOMTs) and smart contracts like MultiSigC and DAOFLC demonstrates the framework's adaptability and versatility. Its decentralized governance structures, involving various stakeholders and validation mechanisms, provide a transparent and democratic framework for managing FL processes. The qualitative evaluation under different threat models showcases 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 underscore the framework's potential impact and scalability. In essence, DAO-FL is a robust solution that strengthens FL integrity through decentralized decision-making and validation mechanisms, setting a new standard for decentralized orchestration in information flows.

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
DAOMTC and DAOC UML Diagram

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

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

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

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

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

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

[26] 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

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

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

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

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

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

[32] “Sepolia Test Network,” Accessed: July 20, 2023. [Online]. Available: https://sepolia.etherscan.io/

[33] Etherscan, “Etherscan APIs,” Accessed: July 20, 2023. [Online]. Available: https://etherscan.io/apis

[34] 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.

[35] 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.