DOI: 10.32604/cmc.2024.050949

ARTICLE

Transparent and Accountable Training Data Sharing in Decentralized Machine Learning Systems

Siwan Noh[1] **and Kyung-Hyune Rhee**[2,*]

[1]Industrial Science Technology Research Center, Pukyong National University, Busan, 48513, South Korea

[2]Division of Computer Engineering, Pukyong National University, Busan, 48513, South Korea

*Corresponding Author: Kyung-Hyune Rhee. Email: khrhee@pknu.ac.kr

Received: 23 February 2024 Accepted: 10 May 2024 Published: 20 June 2024

ABSTRACT

In Decentralized Machine Learning (DML) systems, system participants contribute their resources to assist others in developing machine learning solutions. Identifying malicious contributions in DML systems is challenging, which has led to the exploration of blockchain technology. Blockchain leverages its transparency and immutability to record the provenance and reliability of training data. However, storing massive datasets or implementing model evaluation processes on smart contracts incurs high computational costs. Additionally, current research on preventing malicious contributions in DML systems primarily focuses on protecting models from being exploited by workers who contribute incorrect or misleading data. However, less attention has been paid to the scenario where malicious requesters intentionally manipulate test data during evaluation to gain an unfair advantage. This paper proposes a transparent and accountable training data sharing method that securely shares data among potentially malicious system participants. First, we introduce a blockchain-based DML system architecture that supports secure training data sharing through the IPFS network. Second, we design a blockchain smart contract to transparently split training datasets into training and test datasets, respectively, without involving system participants. Under the system, transparent and accountable training data sharing can be achieved with attribute-based proxy re-encryption. We demonstrate the security analysis for the system, and conduct experiments on the Ethereum and IPFS platforms to show the feasibility and practicality of the system.

KEYWORDS

Decentralized machine learning; data accountability; dataset sharing

1 Introduction

The advent of mobile internet and networking has significantly revolutionized the way we access and utilize information on the move. As mobile device usage continues to increase, so does the demand for faster, more efficient, and more reliable mobile networks. Machine learning (ML), a subset of artificial intelligence (AI), has emerged as a promising solution for improving the performance of mobile networks and enhancing mobile internet security [1]. However, Centralized ML (CML) systems typically involve storing all training data on a single machine. To achieve more precise and enhanced results, these systems require large, high-quality, and diverse datasets collected from various users. This necessitates a significant investment in secure data processing infrastructure, as sensitive personal data is often involved. Additionally, the centralized approach results in substantial costs for the computing power required to train models and the storage space needed to hold datasets.

Decentralized Machine Learning (DML) systems leverage the power of geographically distributed devices to tackle large-scale, complex tasks. This approach harnesses the power of crowdsourcing, where a vast network of participants contributes to the development of AI services. DML's numerous advantages, including increased efficiency [2,3] and improved quality of work [4–6], are driving its adoption across various machine learning domains. Federated Learning [2] divides complex training tasks from a central server and distributes them to networked worker devices. Workers train the pre-distributed model on their local data and send updated model parameters back to the requester. The requester aggregates these updates to build a new, improved machine learning model. Golem Network [3] is a decentralized platform that allows users to rent out their computing resources for various tasks, including machine learning. This significantly reduces training time, enabling the development of more complex models and the utilization of larger datasets. Data labeling, the process of adding informative labels to raw data, plays a crucial role in machine learning model performance. Crowdsourcing platforms like CrowdWorks [4], Appen [5], and Labelbox [6] facilitate diverse data labeling tasks. Participants receive raw data (images, text, videos) and contribute by identifying and assigning meaningful labels, enriching the data with context. In DML systems, anyone can participate as a requester, contributing to model development, or as a worker, offering their resources [7]. This flexibility fosters access to diverse datasets and empowers individuals to develop models. However, it also presents challenges that need to be addressed.

Incentives are crucial in many systems to encourage participation and ensure high-quality contributions. Various incentive mechanisms can be implemented, such as Monetary rewards, Reputation systems, Gamification [8]. In an ideal scenario with honest participants, all workers would provide their resources, and requesters would accurately evaluate workers' contributions to determine fair rewards. However, in a real-world system where participants are not guaranteed to be honest, both requester and worker might act maliciously to maximize their own benefits [9]. In traditional CML systems, developers have full control over all data and computing resources used for development. However, in DML systems, control over these resources is distributed, making it possible for malicious participation to exploit this asymmetry of power and information to their advantage. For example, malicious workers can intentionally submit incorrect data to degrade model performance or perform free-riding attacks [10] by benefiting from the system without contributing. To address these challenges, recent proposals leverage blockchain technology to evaluate and record worker contributions throughout the development process [11–13], ensuring accountability [14–16] and providing resistance against such attacks [17–20].

While malicious workers on platforms have received significant attention for exploiting workers and manipulating data, malicious requesters pose a hidden threat that deserves equal consideration. Amazon Mechanical Turk (MTurk) [21] is a prime example of a crowdsourcing platform vulnerable to both malicious worker and malicious requesters. MTurk is a crowdsourcing platform that allows businesses to outsource tasks to a global workforce of remote workers. Requesters can post tasks, called Human Intelligence Tasks (HITs), on the platform, and workers can complete these tasks for a fee. HITs can be a variety of tasks, such as data entry, image tagging, and machine learning. Despite safeguards, MTurk has seen documented cases of abuse, highlighting the potential dangers posed by malicious participants: 1) *Task Payment Avoidance*: Requesters refusing to pay for completed tasks, essentially stealing workers' effort. 2) *Data Collection without Consent*: Tasks may deceptively collect workers' personal information, raising privacy concerns. 3) *Unfairly Low Pay*: Some tasks offer unreasonably low compensation, exploiting workers and undermining fair labor practices. 4) *Lack of Transparency*: Unclear or misleading task descriptions and requirements hinder informed decision-making by workers. Not only on MTurk, but also on other crowdsourcing platforms, similar ethical issues arise [22]. These cases highlight the importance of research to address potential risks posed by malicious requesters through technological solutions. While blockchain technology can enable automated task management and prevent worker non-compliance or unauthorized actions, it cannot address unfair low pay issues as long as worker evaluation and reward payments are entirely requester-dependent.

In this paper, we propose a transparent and accountable training data sharing method to address the aforementioned challenges. Our proposed method is designed to securely distribute training data (including untrained ML models, training datasets, and test datasets) between potentially malicious requesters and workers within the DML system. First, we introduce a method to transparently split original datasets into training and test datasets using a blockchain smart contract, preventing manipulation by malicious participant. Second, we carefully design the communication protocol for the requester, worker, and storage manager to reduce on-chain communication costs by using content identifiers (CIDs) in the InterPlanetary File System (IPFS) to reference the data instead of handling large datasets directly. Third, we leverage attribute-based proxy re-encryption (ABPRE) to split randomly the training dataset while preventing unauthorized access from malicious participants. This paper makes the following contributions:

- We propose a transparent dataset split method that prevents malicious requester from using inadequate test datasets for evaluation, thereby undermining the value of honest contributions.
- We only record the location of the IPFS data on the blockchain, resulting in a significant reduction in on-chain storage compared with existing works.
- We leverage ABPRE to achieve secure dataset encryption, split, and decryption. ABPRE prevents against malicious requester altering test data before encryption and unauthorized access by malicious workers after splitting.

The remainder of this paper is organized as follows: We investigate related work in Section 2. In Section 3, we provide the system model and security model of the proposed system, and present security goals. Preliminaries and detailed constructions are presented in Sections 4 and 5, respectively. We demonstrate security analysis in Section 6 and performance evaluation of the proposed system in Section 7. Section 8 discusses the benefit of our work and our future work, and Section 9 concludes this paper.

2 Related Work

2.1 Malicious Worker Issue

The U.S. Government Accountability Office (GAO) released an accountability framework (GAO-21-519SP) [23] to help system managers ensure accountable and responsible use of AI in government programs and processes. This framework recommends that system managers should document five key practices (sources, reliability, categorization, variable selection, and enhancement) throughout the development and operation of AI systems to assess the accountability of the datasets used. Despite the GAO framework, identifying unreliable or malicious contributions in DML systems remains a significant challenge. Selecting the right methods and mechanisms to manage data quality is essential for ensuring the accountability of these systems. Transparency and immutability are critical in ensuring the reliability and trustworthiness of DML systems. Blockchain technology, with its inherent features, presents a promising solution to address these requirements. Recent works leveraging blockchain to ensure the accountability of DML systems can be categorized into two main groups based on their focus: Data provenance and data reliability.

The data provenance model leverages the blockchain's immutability to prevent modification of recorded data by attackers. Consequently, many works record the provenance of training data used for model development through blockchain. Typically, hash values of training data are stored on the blockchain [11–13]. For example, Desai et al. [12] and Lo et al. [13] proposed systems that record hash values of training data (e.g., global and local models) for accountable federated learning. These systems enable auditors to obtain a transparent and reliable development history. However, manually adding data to the blockchain carries manipulation risks [24]. Therefore, some suggest a smart contract-based approach to automate logging every stage of the development process without human intervention [14–16]. For instance, Ma et al. [14] and Nguyen et al. [15] proposed a blockchain-assisted decentralized federated learning framework where each client's task, participation, and contributions are managed through a smart contract. Salim et al. [16] introduced a secure Electronic Health Records (EHR) sharing scheme using decentralized learning. Hospitals train local models on patient EHRs stored in the private IPFS network to protect privacy. These models are then aggregated at a research center to create a global model. The results are stored public IPFS while access to the original EHR data remains controlled by patients through smart contracts on a blockchain. However, this approach has the limitation that it can only be identified after a malicious worker has negatively affected the performance of the machine learning model. This approach allows recording a wider range of data, including not only the source of training data but also all framework-generated data. The data provenance model aims to detect or prevent arbitrary modifications of the development history by recording verifiable values of training data in a transparent and accountable database.

In the data reliability model, data that is sufficiently complete, accurate, and applicable for auditing purposes is considered "reliable and trustworthy" [23]. This means that to develop a trustworthy ML model, the completeness, accuracy, and applicability of the training data and its impact on the model must be evaluated and recorded. Model evaluation [25] is a quantitative measure of how well the ML model has been trained, typically assessed by how well it predicts for a given input. One measure is the loss function, a mathematical metric that quantifies the difference between the model's prediction and the actual value [26]. This method can evaluate how closely the model's predictions align with the actual values for a specific test dataset. Solutions within this approach typically implement all training operations on a blockchain smart contract with an incentive mechanism to encourage active worker participation in training. The evaluation results provided by a chosen evaluator [7,17,18,20] or smart contracts [19] are recorded on the blockchain with various information to assess worker contributions and determine individual rewards based on the evaluation result stored in the block.

Blockchain smart contracts can identify malicious client contributions in DML systems. These contracts inspect recorded data to detect tampering and evaluate model improvements before aggregation [12,13,17,18]. This transparency in contribution evaluation enhances the system's reliability [20]. However, challenges persist. While auditors can assess training data, evaluating contributions can be delegated to potentially biased evaluators or rely on outdated datasets chosen by malicious requesters aiming to save incentives. This lack of accountability for the evaluation data itself undermines the transparency and reliability of the entire system, potentially leading to reduced participation, lower performance, and ethical concerns about fairness.

2.2 Malicious Requester Issue

While existing solutions utilizing an on-chain approach are practical for small-scale storage or simple operations, they become impractical for large-scale storage or complex operations. Storing large datasets or implementing client contribution evaluation in a smart contract can significantly increase the computational cost for the blockchain network, resulting in higher transaction fees for users. Existing solutions primarily focus on ensuring worker accountability to prevent malicious contributions, neglecting the threat of malicious requesters who can manipulate the system for their own benefit.

In [22], Xie et al. analyzed the dark side of crowdsourcing platforms by investigating crowdsourcing tasks from multiple micro-task crowdsourcing marketplaces. Micro-task crowdsourcing is a process by which requesters (Businesses) outsource simple, repetitive tasks to a large online workforce. Requesters post tasks on crowdsourcing platforms, and workers can choose the tasks they want to complete and receive payment for their work. Xie et al. demonstrated that ethical issues by platform participants cannot be ignored, and highlighted the issue of requesters with high task authority refusing to pay workers as a major example. Requesters may reject payment for completed tasks for malicious reasons or due to issues such as badly-designed tasks, technical errors, interface design errors, and error-prone evaluation systems. A malicious requester can intentionally obfuscate tasks, causing workers to spend significantly more time and effort than the offered compensation warrants. This can lead to increased task failure rates and payment rejection rates [27,28]. While communication between requesters and workers and reputation-based evaluation systems are proposed as solutions [29], the current DML approach of relying on requester-provided data for task evaluation can fail to distinguish between sincere and malicious worker contributions. For example, a malicious requester might incentivize users to participate in model development by measuring their contribution based on improvements in the global model performance. Just as workers can use misleading data during training, malicious requesters can manipulate the test dataset used for evaluation. While providing machine images as evaluation data for a model trained on animal images can be helpful for assessing generalization performance, it cannot accurately evaluate the model's primary function of animal image classification, and thus cannot be considered an accurate evaluation. Therefore, ensuring a trustworthy DML system requires a recording procedure not only the training data (to address malicious clients) but also the test data used in evaluation (to address malicious parameter servers). This approach ensures transparency in the evaluation of task results, which previously relied on requesters, thereby enabling the identification of malicious requesters.

Compared to existing works, our system, as shown in Table 1, fulfills the GAO requirements while addressing malicious requester and worker issues under a practical and reasonable DML model. We specifically tackle the accountability challenges related to the evaluation dataset.

Table 1: Summary of blockchain-based DML system

	Training data's			Security requirement	
	Reliability	Provenance	Confidentiality	Malicious requester	Malicious worker
[7]	Model evaluation	On-Chain	Symmetric encryption	N/A	N/A
[12]	Model evaluation	On-Chain	TLS/SSL	N/A	✓

(Continued)

Table 1 (continued)

	Training data's			Security requirement	
	Reliability	Provenance	Confidentiality	Malicious requester	Malicious worker
[13]	Model evaluation	On/Off-Chain	Asymmetric & symmetric encryption	N/A	N/A
[16]	N/A	On/Off-Chain	Private IPFS network	N/A	N/A
[17]	Model evaluation	On-Chain	N/A	N/A	N/A
[18]	Model evaluation	On/Off-Chain	Elliptic-curve Diffie–Hellman encryption & differential privacy	N/A	✓
[19]	Model evaluation	On-Chain	N/A	✓	✓
[20]	Model evaluation	On-Chain	N/A	N/A	N/A
Proposed	Model evaluation	On/Off-Chain	Attribute-based proxy re-encryption	✓	✓

3 Problem Statement

3.1 System Model

Our system includes the following entities: Supervising Authority (SA), Requester (RQ), Worker (W), Evaluation Node (EV), Gateway Server (GS) and smart contracts.

- SA is responsible for supervising the DML system and issuing cryptographic keys based on participant roles.
- RQ refers to users who want to develop ML models on the DML system. RQ can have enough training datasets to train ML models but much fewer computing resources than a powerful server. RQ deploys a project for ML model development on the blockchain and distributes training dataset to be used in this project through smart contracts.
- W is a worker that is interested in the incentives of ML model development projects. For example, W can be an IT company that can provide idle computing resources. W participates in the project deployed by RQ and receives rewards for their contributions.
- EV evaluates the contributions of W and reports the results to the corresponding RQ. We assume that our system has a group of evaluators managed by SA. This group has a group leader EV_L that acts as an intermediary between W and a member of EV to efficiently spread messages within the group.
- GS is a node in the IPFS network. All system entities can retrieve a training dataset stored in the IPFS network through this server. GS is responsible for monitoring all smart contracts deployed by RQ. GS distributes the training and evaluation datasets to the project's participating Ws and EVs after RQ uploads the training dataset to the IPFS network.
- In the system, smart contracts support project creation, management, dataset shuffling, and distribution. In the proposed communication protocol, these primary functions are implemented using two smart contracts: 1) Project Creation and Management Contract ($PCMC$), and 2) Dataset Shuffling and Distribution Contract ($DSDC$).

Fig. 1 illustrates the proposed system's four phases. (1) Project Setup: RQ defines project parameters (e.g., rewards, goals, worker requirements) and deploys smart contracts on the blockchain network to initiate a new project. Worker candidates applying to participate are verified by RQ against project requirements (e.g., deposit, computing power [30], reputation [31]). Only qualified workers are selected for the project. (2) Project Preparation: RQ encrypts the dataset with a secret key issued by SA and uploads the encrypted datasets to the IPFS network. This secret key is unique and generated based on the RQ's identifiable attributes. (3) Dataset Split: $PCMC$ and $DSDC$ collaboratively split the dataset into training and test datasets using a two-part split method. (4) Dataset Distribution: GS retrieves the dataset from the IPFS network and distributes a re-encrypted training and test datasets along with machine learning model. The training datasets are sent to W, while the test datasets are sent to EV_L. After training, W submit their trained local models to EV. EV then broadcasts these submitted learning results to all members within EV group. Finally, EV evaluates the ML models using the provided test datasets and reports the results to $PCMC$. Once all procedures are complete, RQ can access the learning results of each worker and their corresponding evaluations through $PCMC$.

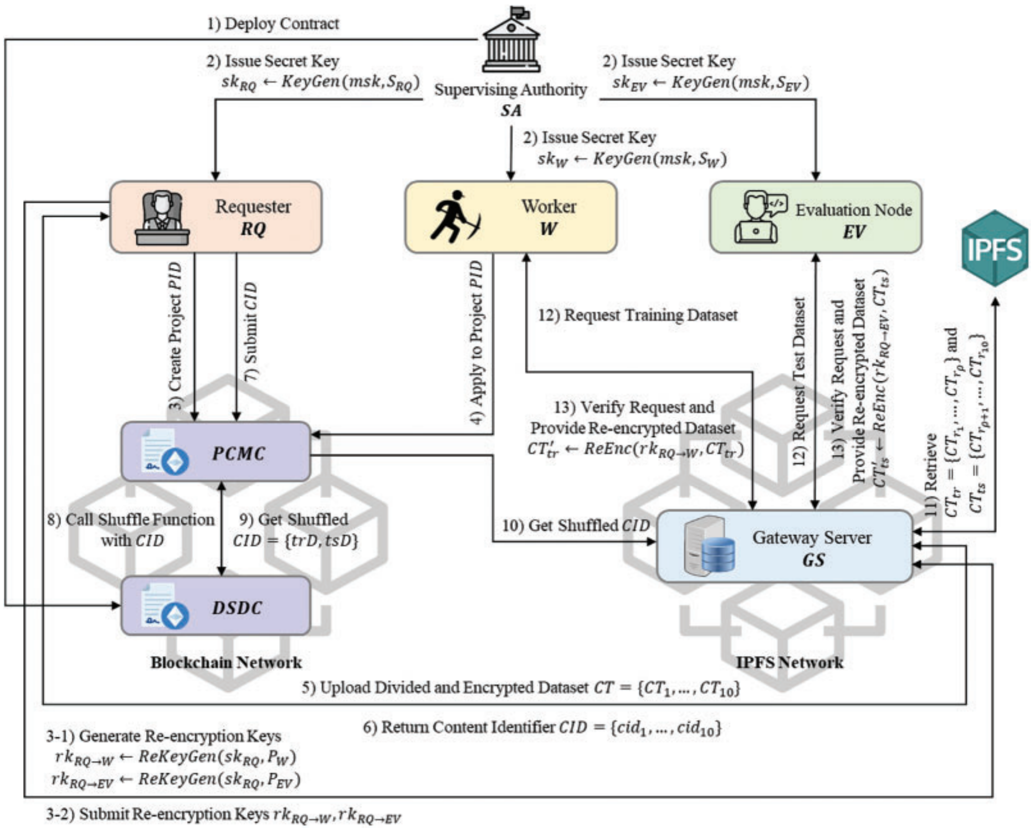

1) Deploy Contract
2) Issue Secret Key $sk_{RQ} \leftarrow KeyGen(msk, S_{RQ})$
Supervising Authority SA
2) Issue Secret Key $sk_{EV} \leftarrow KeyGen(msk, S_{EV})$
2) Issue Secret Key $sk_W \leftarrow KeyGen(msk, S_W)$
Requester RQ
Worker W
Evaluation Node EV
IPFS
3) Create Project PID
7) Submit CID
4) Apply to Project PID
12) Request Training Dataset
12) Request Test Dataset
13) Verify Request and Provide Re-encrypted Dataset $CT'_{ts} \leftarrow ReEnc(rk_{RQ\to EV}, CT_{ts})$
11) Retrieve $CT_{tr} = \{CT_{r_1}, \dots, CT_{r_p}\}$ and $CT_{ts} = \{CT_{r_{p+1}}, \dots, CT_{r_{10}}\}$
13) Verify Request and Provide Re-encrypted Dataset $CT'_{tr} \leftarrow ReEnc(rk_{RQ\to W}, CT_{tr})$
PCMC
8) Call Shuffle Function with CID
9) Get Shuffled $CID = \{trD, tsD\}$
10) Get Shuffled CID
Gateway Server GS
DSDC
Blockchain Network
IPFS Network
5) Upload Divided and Encrypted Dataset $CT = \{CT_1, \dots, CT_{10}\}$
6) Return Content Identifier $CID = \{cid_1, \dots, cid_{10}\}$
3-1) Generate Re-encryption Keys $rk_{RQ\to W} \leftarrow ReKeyGen(sk_{RQ}, P_W)$ $rk_{RQ\to EV} \leftarrow ReKeyGen(sk_{RQ}, P_{EV})$
3-2) Submit Re-encryption Keys $rk_{RQ\to W}, rk_{RQ\to EV}$

Figure 1: Proposed system overview

3.2 *Security Model and Goals*

Homo economicus [9] is an economic model that assumes humans always act rationally and aim to maximize their own utility. This theory can be applied to crowdsourcing systems to predict the behavior of requesters and workers. RQ aims to get the best results at the lowest cost and W tries to complete tasks quickly and efficiently to earn more money. Therefore, our system design considers the following malicious behaviors as security threats that need to be addressed:

- *Malicious Requester*: To minimize development costs, a malicious requester might attempt to undermine the contributions of honest workers. Workers submits their trained local models, each designed to perform well on the training data it was trained on. However, evaluating model performance requires unseen test datasets that are large, diverse, and unknown to the model beforehand [32]. In machine learning, a dataset consists of various features (measurable data points) with corresponding correct answers (labels) within a specific value range (sample space). For instance, the MNIST dataset contains handwritten digits (0–9) as images with 784 pixel values (0–255). If a malicious requester controls the test dataset, they could introduce fake data (data outside the training dataset's features or sample space) to manipulate the evaluation process and unfairly devalue the workers' contributions by generating inaccurate model performance results.
- *Malicious Worker*: While malicious workers can launch both untargeted and targeted adversarial attacks [12,33,34], specifically targeting individual models within small collaborative teams provides minimal benefit in our system. Therefore, the primary concern is the untargeted "lazy node" attack [14], which the requester can address by evaluating local model accuracy and withholding rewards accordingly. This paper tackles a more challenging scenario: A malicious worker who bypasses evaluation with minimal effort by generating an overfitting model using the test dataset beforehand [19]. Overfitting describes a model that becomes overly reliant on the training data, hindering its ability to generalize to unseen data. If a malicious worker has access to the test data before training, they can create a model that performs well on the test set but poorly when integrated into the final global model.

Under the system and security models, we aim to achieve the following security goals:

- *Unpredictable Dataset Split*: This ensures two key aspects: (1) Transparency and reliability in the dataset splitting process; (2) Unpredictable and unalterable split results for all participants in the system.
- *Confidentiality of Test Dataset*: This guarantees two critical points: (1) The test dataset remains inaccessible to all workers; (2) Even if a malicious worker intercepts the test dataset through network eavesdropping, they cannot obtain the original, unencrypted data (plaintext).

To address security concerns arising from potential malicious users acting as Requesters or Workers, we propose a secure dataset sharing method for DML systems. To achieve transparency and reliability in dataset splitting, we leverage a blockchain smart contract. Additionally, we utilize ABPRE to guarantee data confidentiality during communication between Requesters and Workers over the internet.

4 Preliminaries

4.1 Attribute-Based Proxy Re-Encryption

The ABPRE [35–37] is a useful cryptographic scheme that combines attribute-based encryption and proxy re-encryption. This scheme allows a semi-trusted proxy to transform a ciphertext under an access policy into a re-encrypted ciphertext under a designated access policy without revealing any information related to the ciphertext. The ABPRE scheme comprises the following six algorithms:

- $(param, msk) \leftarrow Setup(\lambda, U)$: The system manager executes *Setup* with a security parameter λ and the attribute universe U as its inputs to generate public parameters *param* for the system, and then generates its master secret key *msk*.
- $sk_S \leftarrow KeyGen(msk, S)$: The system manager validates the user attribute set S and executes *KeyGen* with its master secret key *msk* and attribute set S as its inputs. This algorithm outputs the secret keys sk_S for system participants.
- $CT \leftarrow Enc(m, P)$: This algorithm encrypts a message m with an access policy P and generates a ciphertext CT (where P represents the access structure $(\mathbb{A}, \rho)$, $\mathbb{A}$ is an $l \times n$ matrix, and a function $\rho(j) \in \mathcal{P}, j \in \{1, \ldots, l\}$).
- $rk_{S \to P'} \leftarrow ReKeyGen(sk, P')$: This algorithm generates a re-encryption key $rk_{S \to P'}$ which can be used to transform a ciphertext under P to another ciphertext under P' (where P and P' are two disjoint access structures).
- $CT_R \leftarrow ReEnc(rk_{S \to P'}, CT)$: This algorithm transforms the original ciphertext CT into a re-encrypted ciphertext CT_R using the re-encryption key $rk_{S \to P'}$.
- $m \leftarrow Dec(sk_S, CT)$, $m \leftarrow Dec_R(sk_S, CT_R)$: This algorithm decrypts the encrypted ciphertext under P and P' using the corresponding secret key, respectively.

To ensure the confidentiality of the test datasets in the dataset distribution process, we utilize the ABPRE scheme. After splitting the dataset, the encrypted dataset under *RQ*'s access policy is transformed into a ciphertext under the corresponding participant's access policy on the semi-trusted IPFS server.

4.2 Blockchain Smart Contract

Blockchain smart contracts are self-executing programs stored on a decentralized ledger known as a blockchain. These programs automate the performance of a contract when predefined conditions are met, eliminating the need for intermediaries like lawyers or escrow services. Smart contracts offer several advantages, including:

- *Reduced Trust Dependency*: By relying on cryptographic verification and tamper-proof blockchain records, smart contracts remove the need for trusted third parties to enforce agreements, minimizing potential manipulation or fraud.
- *Enhanced Efficiency*: Automating contract execution streamlines workflows, eliminates manual processing delays, and reduces overall transaction costs.
- *Increased Transparency*: All participants in a smart contract can access and verify its terms and execution history on the blockchain, fostering trust and accountability.

This paper leverages the Ethereum public blockchain to guarantee the integrity of our dataset split results. Smart contracts are distributed and executed independently across the network's nodes, with the final outcome confirmed through a consensus mechanism among these nodes. To facilitate stateful smart contracts, which can track and respond to changes over time, blockchain nodes permanently store the current state of each contract.

However, it is crucial to recognize the resource limitations associated with complex smart contract operations. These processes consume computational power at each node, potentially raising transaction fees. To optimize a blockchain-based system, it is essential to design it in a way that minimizes on-chain computations and data processing, focusing only on essential elements that require the blockchain's security guarantees.

4.3 IPFS Storage

The IPFS storage [38] is a peer-to-peer (P2P) hypermedia protocol that revolutionizes data storage by enabling a decentralized network. Unlike traditional web protocols reliant on location-based addressing, IPFS employs a content-based addressing system. This system leverages unique cryptographic hashes called CIDs derived from the data itself. Combined with the Kademlia routing algorithm [39], CIDs facilitate efficient data retrieval across the distributed IPFS network.

During project preparation, datasets are split into smaller fragments and distributed strategically across a network of IPFS nodes according to predefined rules. This distributed storage approach ensures redundancy, where multiple copies of each data block reside on various nodes. Even if individual nodes become unavailable, the data remains accessible because users can retrieve the desired block using its unique CID from any available node hosting that specific fragment.

In our system, all datasets are encrypted for secure transfer over the IPFS network between requesters and worker nodes. However, assuming all participants are native IPFS nodes might not be feasible. Therefore, we leverage a semi-trusted IPFS node to act as a gateway, facilitating access to the decentralized storage network for other entities in the system. Ideally, a network of multiple gateway servers would be implemented to eliminate single points of failure. However, for demonstration purposes, this system is simplified to utilize a single gateway server.

5 Transparent and Accountable Training Data Sharing

5.1 Project Setup

We assume that supervising authority SA has executed $Setup()$ to complete the system initialization, and that each participant has obtained a secret key $sk \leftarrow KeyGen(msk, S)$ for their given role (i.e., requester RQ, worker W, and evaluation node EV). The attributes set S represents the role of the participant in the system. The secret keys of W and EV indicate their roles (S_W and S_{EV}, respectively), whereas the secret keys of RQ are all generated from uniquely identified attributes S_{RQ_j}. In other words, W and EV who play the same role have the same decryption capability, whereas each RQ has a unique encryption capability in the system. Thus, RQ must be certified by SA to have sufficient funds to play the role. As shown in Fig. 2, RQ creates a new project and recruits W as follows:

1) RQ deploys the Project Creation and Management Contract ($PCMC$) that includes the project description, the required deposit D, the project rewards R, the number of workers N, and the required computing power ($time$, $target$) as project parameters.
2) Candidate worker W places deposit D_W on $PCMC$ and applies to the project.
3) $PCMC$ evaluates W's computing power using Algorithm 1. If returns $True$, each candidate's ID is recorded, else deposit D_W is deposited.
4) If the number of candidates who passed the test achieves goal N, $PCMC$ determines the list of workers $list_{PID} = \{W_1, \ldots, W_j\}$ to participate in the project PID.

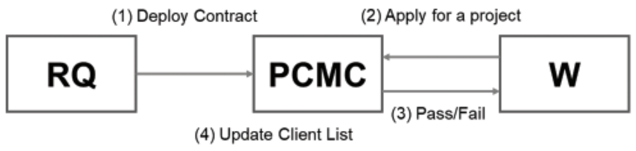


Figure 2: Overview of project setup

```
Algorithm 1: evaluateComputingPower
input: $D$: a required deposit, $D_W$: a deposit of the candidate $W$, $target$: the number of leading zeros,
$tTime$: time threshold, $nonce$: A solution, $cTime$: a cryptographic salt given to $W$
output: True/False
1. $PCMC$ check amount of deposit $D_W$
    if $D_W < D$, throws;
    else return $cTime \leftarrow$ currentTime();
2. $W$ finds $nonce$ and submits it to $PCMC$
3. $PCMC$ compute $result \leftarrow Hash(cTime, nonce)$
4. if $(result < target) \bigwedge ((currentTime() - cTime) < tTime)$
    return True;
  else return False;
5. End
```

5.2 Project Preparation

After the Project Setup phase is completed, requester RQ distributes his datasets to workers W and evaluation nodes EV through project management contract $PCMC$. The traditional approach uploads the datasets to the storage of the smart contract and splits them within the contract. However, this method consumes a significant amount of resources in the blockchain network. Moreover, in the context of dataset split for training and test purposes, maintaining an accurate representation of the data distribution is crucial for reliable model evaluation. If the data distribution differs between the training and test datasets, the model's ability to generalize to unseen data may be compromised.

To address these issues and conserve the storage space required for split, we store CIDs linked with the datasets instead of the datasets themselves and use the stratified k-fold cross-validation (SKCV) method [40] to divide dataset into multiple data blocks with an even distribution. Regardless of the data volume stored in the IPFS network, all CIDs have the same length of 256 bits.

As shown in Fig. 3, RQ divides the original dataset $Data_{RQ}$ into multiple blocks and encrypts each block using his secret key sk_{RQ}. Subsequently, RQ uploads these blocks to the IPFS network through gateway server GS and submits the CIDs returned from the IPFS network to $PCMC$ as shown in Fig. 4. $PCMC$ then stores these CIDs in an array, which is a fixed-size sequential collection of elements of the same type. A detailed procedure is as follows:

1) RQ determines an index ρ (where $1 \leq \rho \leq 9$) and divides the original dataset $Data_{RQ}$ into 10 equal-sized parts $Data_{RQ} = \{block_1, \ldots, block_{10}\}$.
2) RQ encrypts each block under his secret key sk_{RQ} and generates ciphertext $CT_i \leftarrow Enc(block_i, P_{RQ})$ for each $block_i \in Data_{RQ}$.
3) RQ uploads $CT = \{CT_1, \ldots, CT_{10}\}$ to the IPFS network through GS, and obtains $CID = \{cid_1, \ldots, cid_{10}\}$, where cid_i is a content identifier for retrieving CT_i from the IPFS network.
4) RQ submits these CIDs to $PCMC$.

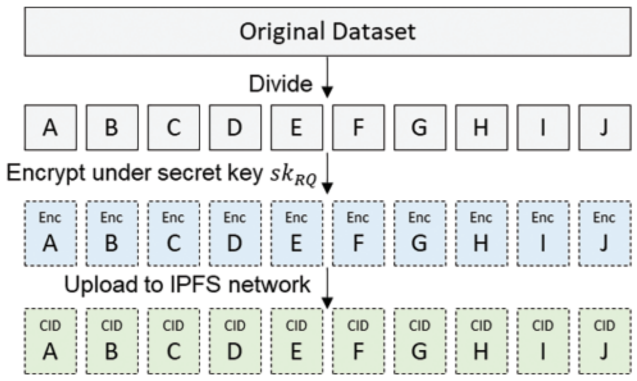

Original Dataset
Divide
A B C D E F G H I J
Encrypt under secret key sk_{RQ}
Enc A Enc B Enc C Enc D Enc E Enc F Enc G Enc H Enc I Enc J
Upload to IPFS network
CID A CID B CID C CID D CID E CID F CID G CID H CID I CID J

Figure 3: Dataset divide and encrypt

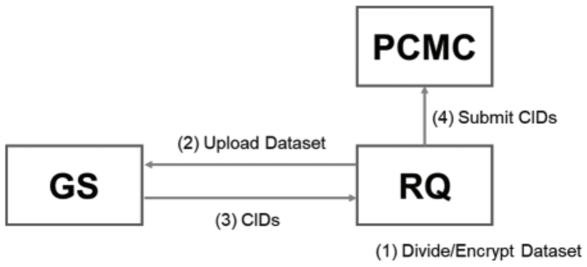


Figure 4: Overview of project preparation

5.3 Dataset Split

In this phase, the project management contract $PCMC$ splits the original dataset $Data_{RQ}$ into a training dataset for workers W and a test dataset for evaluation nodes EV on behalf of the requester RQ. To distribute the datasets, $PCMC$ calls a shuffle function in the dataset distribution contract $DSDC$ with CIDs. As shown in Fig. 5, $DSDC$ randomly sorts CIDs and assigns the former index number ρ to the training dataset, and the latter index number to the test dataset. An overview of this process is shown in Fig. 6, and a detailed procedure is as follows:

1) $PCMC$ records the CIDs as $CID_{RQ} = \{PID, CID, \rho, trD, tsD\}$, where PID is a contract address, and trD and tsD are empty arrays for the training and test datasets, respectively.
2) RQ calls the shuffle function in $DSDC$ with an array of CIDs via $PCMC$.
3) $DSDC$ shuffles CIDs and returns a shuffled CIDs using Algorithm 2.
4) $PCMC$ sets the shuffled CIDs at trD and tsD, respectively, based on ρ.

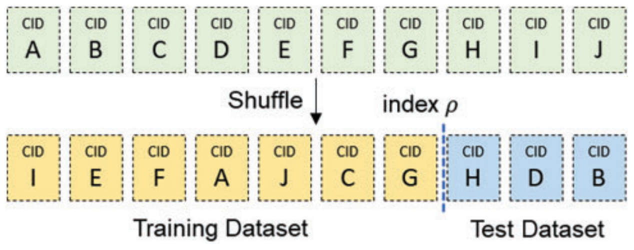

CID A CID B CID C CID D CID E CID F CID G CID H CID I CID J
Shuffle
index ρ
CID I CID E CID F CID A CID J CID C CID G CID H CID D CID B
Training Dataset
Test Dataset

Figure 5: Dataset shuffle

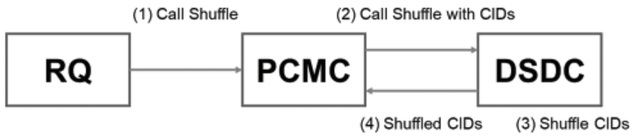


Figure 6: Overview of dataset split

Algorithm 2: datasetShuffle

```
input: $CID = \{cid_1, \ldots, cid_n\}$: an array of $n$ elements, $trD, tsD$: an empty array for the assigned dataset,
$\rho$: an index to split the original dataset into a training dataset and a test dataset
output: a shuffled and split array $trD, tsD$
1. shuffle array $CID$
     For $i = CID.length - 1; i > 2; i-$ do
       $j$ = random integer such that $0 \le j \le i$
       swap $CID[i]$ with $CID[j]$
     return shuffled $CID = \{cid_{r_1}, \ldots, cid_{r_{10}}\}$
2. set shuffled $CID$ to empty array $trnD, tstD$
    For $i = 0; i < CID.length; i{+}{+}$ do
      if $(i > \rho)$ $trD[i] = CID[i]$
      else $tsD[i - (trD.length)] = CID[i]$
    return $trD, tsD$
3. End
```

5.4 Dataset Distribution

After completing Dataset Split phase, the split result becomes visible to all participants in the system. However, the datasets are encrypted under the secret key of the requester RQ. Therefore, to enable workers W and evaluation nodes EV to access this dataset, the encrypted dataset must be transformed such that it can be decrypted using its decryption capabilities. The following describes how RQ generates a re-encryption key and shares it with the gateway server GS.

1) RQ generates the re-encryption keys $rk_{RQ\to W} \leftarrow ReKeyGen(sk_{RQ}, P_W)$ and $rk_{RQ\to EV} \leftarrow ReKeyGen(sk_{RQ}, P_{EV})$ using his secret key sk_{RQ} and the access policies P_W and P_{EV}, respectively.
2) RQ sends the re-encryption keys $rk_{RQ\to W}$ and $rk_{RQ\to EV}$ to GS.
3) GS keeps $< PID, rk_{RQ\to W}, rk_{RQ\to EV} >$.

In Section 3.1, we assumed that GS monitors all projects deployed in the system. For projects that have completed the dataset split phase, As shown in Fig. 7, GS retrieves the encrypted dataset blocks from the IPFS network and transforms each block using a corresponding re-encryption key provided by RQ according to the split results. When project participants request GS for the distribution of the dataset, GS verifies the request and returns the re-encrypted datasets assigned to the role of the requester as follows:

1) GS gets $CID_{RQ} = \{PID, CID, \rho, trD, tsD\}$ by calling a getter function in $PCMC$, where $trD = \left\{cid_{r_1}, \ldots, cid_{r_\rho}\right\}, tsD = \left\{cid_{r_{\rho+1}}, \ldots, cid_{r_{10}}\right\}$.
2) GS retrieves the original ciphertext $CT_{tr} = \{CT_{r_1}, \ldots, CT_{r_\rho}\}$ and $CT_{ts} = \{CT_{r_{\rho+1}}, \ldots, CT_{r_{10}}\}$ encrypted under the RQ's secret key from the IPFS network using trD and tsD, respectively.
3) GS generates re-encrypted ciphertext $CT'_{tr} \leftarrow ReEnc(rk_{RQ\to W}, CT_{r_i})$ for each $CT_{r_i} \in CT_{tr}$ and $CT'_{ts} \leftarrow ReEnc(rk_{RQ\to EV}, CT_{r_i})$ for each $CT_{r_i} \in CT_{ts}$.

4) W and EV requests GS to provide datasets for their project operations.
5) GS verifies whether the requester is a participant in the project via $PCMC$.
6) If the request is valid, GS provides the corresponding re-encrypted ciphertext.
7) W decrypts the re-encrypted ciphertext $data_{r_i} \leftarrow Dec(sk_W, CT_{r_i})$ for each $CT_{r_i} \in CT'_{tr}$ and obtains the training dataset $Data_{tr} = \{data_{r_1}, \ldots, data_{r_\rho}\}$.
8) EV decrypts the re-encrypted ciphertext $data_{r_i} \leftarrow Dec(sk_{EV}, CT_{r_i})$ for each $CT_{r_i} \in CT'_{ts}$ and gets the test dataset $Data_{ts} = \{data_{r_{\rho+1}}, \ldots, data_{r_{10}}\}$.

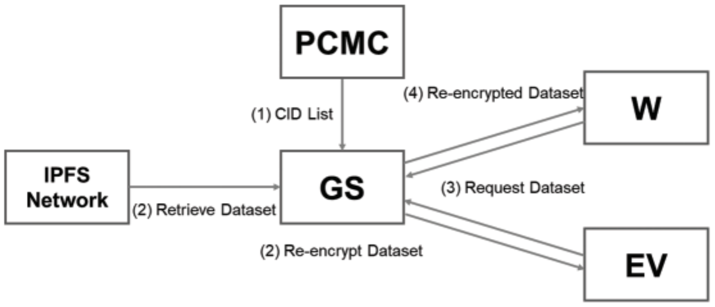


Figure 7: Overview of dataset distribution

6 Security Analysis

6.1 Unpredictable Dataset Split

In the DML system, worker rewards are based on their local models' contribution to the global model. However, a malicious requester *RQ* can unfairly manipulate rewards by evaluating contributions using a misleading test dataset. This dataset, with different features and samples than the training data, provides inaccurate assessments. Our system addresses this by implementing a trusted third-party approach. The smart contract, not *RQ*, splits the original dataset uploaded to the IPFS network. By removing *RQ*'s control over data splitting, providing a fake test dataset becomes significantly more difficult.

Let D be the original dataset uploaded to the IPFS network. *RQ* submits CID, denoted by $cid(D)$, to the smart contract. The immutability of the blockchain guarantees that the data associated with $cid(D)$ remains unchanged after upload. We can express this as:

$$\forall t > upload\_time(D): D(t) = \mathrm{D}(upload\_time\,(D)) \tag{1}$$

where $D(t)$ represents the state of dataset D at time t, and $upload\_time(D)$ denotes the time at which D was uploaded. A successful manipulation attack by *RQ* would require either: 1) Replacing $cid(D)$ with a new CID, $cid(D')$, pointing to a fake dataset D'. 2) Finding a distinct dataset D' such that $cid(D') = cid(D)$.

The collision resistance property of the cryptographic hash function used to generate CIDs makes the second scenario highly improbable. Mathematically, the probability of a collision $P(cid(D') = cid(D))$ can be approximated as negligible for secure hash functions. Furthermore, modifying the blockchain to replace $cid(D)$ with a new CID is computationally infeasible due to the Byzantine Fault Tolerance (BFT) consensus mechanism employed by most blockchain platforms. The BFT protocol ensures that all honest nodes in the network agree on the current state of the blockchain, making it nearly impossible for a malicious actor to tamper with data without detection.

Therefore, by leveraging the immutability of the blockchain and the collision resistance of cryptographic hash functions, our trusted third-party approach demonstrably reduces the probability of a successful manipulation attack by RQ to a negligible value.

6.2 Confidentiality of Test Dataset

DML system faces a challenge in ensuring data confidentiality, particularly for the test dataset used to evaluate workers' contributions. Malicious workers could potentially gain access to the test dataset and manipulate their local models to achieve undeserved rewards or degrade the overall model performance.

The current approach using a test dataset for evaluation is vulnerable to overfitting attacks. If malicious workers obtain the test dataset beforehand (denoted as D_T), they can overfit their local models (denoted as M_W) to achieve high performance on D_T, even if M_W performs poorly on unseen data. This manipulation becomes problematic because in DML, the requester cannot directly access the training data used by malicious workers (denoted as D_{train}).

To address this challenge and ensure data confidentiality, we propose a solution leveraging ABPRE scheme. Let P_{RQ} be the access policy of the requester RQ. The original dataset D is encrypted under P_{RQ} and transformed to a ciphertext CT, denoted as $CT \leftarrow Enc(D, P_{RQ})$, and uploaded to the IPFS network. The Project Creation and Management Contract ($PCMC$) splits CT into training (CT_{tr}) and test datasets (CT_{ts}). The Gateway Server GS then performs proxy re-encryption using ABPRE. CT_{ts} is re-encrypted for Evaluation Node EV under its access structure P_{EV}. This ensures that only evaluation nodes with the corresponding decryption capability derived from sk_{EV} can access CT_{ts}. In the above setting, the security threats of such a system can be categorized by the following two scenarios:

Network Eavesdropping: Even if a malicious worker intercepts the communication between GS and EV over a public network, the re-encrypted test dataset ($CT'_{ts} \leftarrow ReEnc(rk_{RQ\rightarrow EV}, CT_{r_i})$) remains inaccessible due to the inability to decrypt it without the corresponding key derived from sk_{EV}.

Compromised IPFS Node: If a malicious worker attempts to retrieve the test dataset stored on IPFS by becoming an IPFS node, they will still encounter the encryption under sk_{RQ}, rendering the data unreadable.

The security of ABPRE has been extensively studied and proven in prior work [35–37]. In this paper, we focus on the application of ABPRE in the context of DML and demonstrate its benefits for data confidentiality. We leverage ABPRE to address the challenges of overfitting and data leakage in DML, without repeating the security proofs of ABPRE itself.

7 Performance Evaluation

7.1 Complexity Analysis

In the on-chain-based system, the requester RQ stores large datasets directly in the blockchain. Therefore, the average on-chain storage cost for each dataset is $\frac{cost_{store} * |Data|}{256}$, where $cost_{store}$ is the cost of storage in the blockchain network and $|Data|$ is the average size of the dataset. According to the Ethereum yellow paper [41], $cost_{store}$ can be 20,000 gas per 256 bits in Ethereum, considering only the cost of storage space. Suppose that the MNIST-784 dataset is submitted, which results in the use of 76,250,000,000 gas, where the size of the MNIST-784 dataset is 122 MB. In the proposed system, large datasets are stored in the IPFS network and only CIDs are stored in the blockchain, resulting in a total on-chain storage cost of $n * \frac{cost_{store} * |Data|}{256}$, where n is the number of data blocks. In practice, if RQ divides the original dataset into 10 blocks, n can be 10, and $|Data|$ is only 256 bits, which results in 200,000 gas used. In conclusion, the proposed system offers a significantly lower storage cost compared to the on-chain storage model by leveraging IPFS for data storage and storing only CIDs on the blockchain.

7.2 On-Chain Overheads

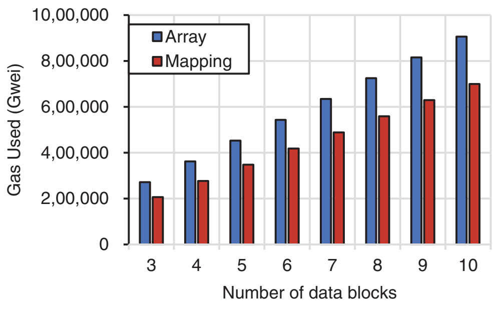
For on-chain overheads, since we only upload CIDs onto the blockchain, the major factor that increases the on-chain overhead is the number of data blocks divided from the original dataset. Therefore, we implement and test the main functions of the dataset distribution contract $DSDC$ on the local Ethereum blockchain. We use the Ganache and Remix IDE to deploy contracts, execute commands, and inspect the states of the contracts.

The storage and splitting of the CIDs are written as a Solidity function. $DSDC$ stores each CID in an array and uses Algorithm 2 to shuffle its elements. Therefore, the number of CIDs stored in a smart contract increases with the number of data blocks derived from the original dataset. Table 2 shows the relationship between the number of data blocks and the gas consumption required to execute each $DSDC$ function and the results of converting the gas consumption of $DSDC$ implemented by the byte array into US dollars. The price of the gas that we set in the test is $2 * 10^{-8}$ ether (20 Gwei), and the cost required to execute the contract is calculated as the multiply of the gas price and the used gas. Even as the number of data blocks increases, the dataset split function consumes relatively little gas. However, the gas consumption for the store CIDs function increases significantly compared to the split function as the number of data blocks increases.

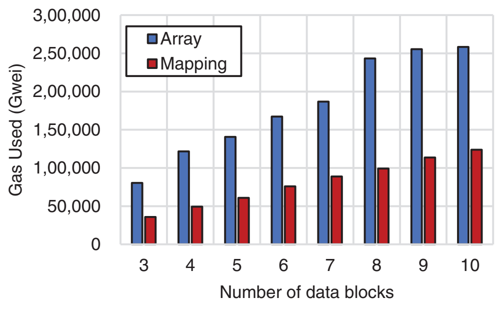
Table 2: Costs required to execute DSDC implemented as an array

Command	Cost	Number of data blocks							
		3	4	5	6	7	8	9	10
Store	Gas	271,881	362,508	453,135	543,762	634,389	725,016	815,643	906,270
CIDs	USD	10.23	13.64	17.04	20.45	23.86	27.27	30.68	34.09
Dataset	Gas	80,451	121,659	140,739	167,210	186,819	243,291	255,508	258,316
Split	USD	3.03	4.58	5.29	6.29	7.03	9.15	9.61	9.72

The cost associated with storing CIDs, as shown in Table 2, is unreasonably high from a practical perspective. However, gas consumption can be significantly reduced by optimizing smart contracts. The gas consumption required to execute a smart contract is determined by many factors such as on-chain data, on-chain operations, and variable orderings, among others. Figs. 8a and 8b show the gas consumption when the main functions (a) store CIDs and (b) dataset split are implemented in an array and mapping type, respectively. The mapping type is similar to the array type in that it is a Solidity reference type meant for storing a group of data. Although only the storage type of CIDs has been changed, the store CIDs function can reduce the cost by 20% and the split function by 50% as shown in Fig. 8a. Moreover, the cost can be reduced to a reasonable level by selecting a CID version that generates CIDs with a shorter length or by changing the cryptographic hash function that creates the CIDs. While not extensively explored here, smart contract optimization techniques present promising avenues for cost reduction. Future work could delve deeper into these optimization strategies.



(a) Store CIDs



(b) Dataset Split

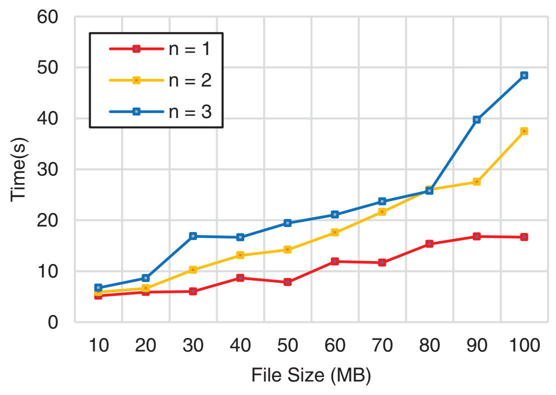
Figure 8: Comparison of on-chain overheads

7.3 Off-Chain Overheads

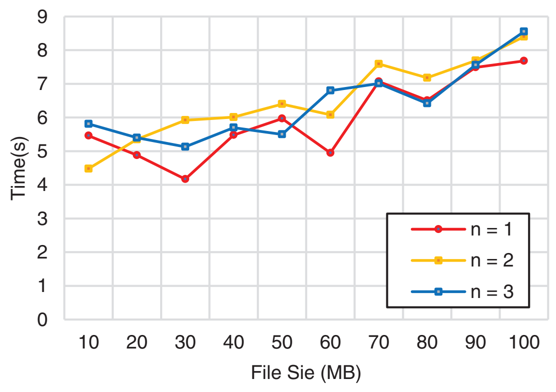
Table 3 represents the off-chain operations required in each phase of the proposed system, along with the entities responsible for performing these operations. In the project preparation, the requester *RQ* uploads an encrypted dataset to the IPFS network via the gateway server *GS*. This dataset is then split on IPFS through on-chain smart contracts and retrieved and re-encrypted by *GS* before being distributed to each participant. As a result, the overhead of *GS* increases linearly with the number of projects existing in the system in Figs. 9a and 9b. We implement and test the overhead for the off-chain operation on a laptop with a 2.30 GHz processor, 8 GB of memory and 1 Gbps network bandwidth. We use the Pinata platform to access data on the IPFS network. We set *n* as the number of users simultaneously requesting operation from the gateway and measured the time it takes to upload and download data to IPFS using dummy data generated with Windows fusutil, varying the size of the dummy data from 10 to 100 MB. The following operations that can significantly affect performance depending on the ABPRE algorithm used in this evaluation were excluded: Secret key generation, Encryption, Decryption, and Re-encryption.

Table 3: Off-chain operations in the proposed system

Phase	Entity	Operation
Project setup	*SA*	Secret key generation
Project preparation	*RQ*	Encrypt dataset
	GS	Upload ciphertext to IPFS
Dataset split	–	–
Dataset distribution	*RQ*	Re-encryption key generation
	GS	Retrieve ciphertext from IPFS
		Transform ciphertext
	EV	Decrypt dataset
	W	Decrypt dataset



(a) Upload



(b) Download

Figure 9: Data upload and download time on IPFS

For the dataset upload, it time around 16 s when $n = 1$ with the 100 MB file size. It took around 37 and 48 s when $n = 2$ and $n = 3$, respectively. As shown in Fig. 8a, the upload time increases linearly with the file size and the number of concurrent requests. On the other hand, the dataset download time was hardly affected by the number of concurrent requests and showed an almost constant time as shown in Fig. 8b. This demonstrates that the overhead of *GS* can significantly increase depending on the number of concurrent requests and the dataset size. However, the impact of this delay on the system security is minimal, and it is possible to reduce the overhead to an acceptable level by deploying multiple interconnected *GS*s within the system.

8 Discussions

8.1 Impacts of Proposed System

In this paper, we propose a novel DML system that addresses the challenges posed by malicious participants aiming to maximize their own benefits in one of the operational modes of DML systems where the requester crowdsources complex machine learning tasks to workers. Our system leverages blockchain, IPFS, and ABPRE to mitigate these challenges and ensure secure and reliable collaboration. Unlike existing approaches where the requester is solely responsible for all task-related procedures, our system minimizes requester involvement by randomly splitting the ABPRE-encrypted dataset using a smart contract. This approach minimizes the potential for malicious behavior by the requester.

Our system offers significant benefits for both workers and requesters within the DML ecosystem, ultimately fostering wider adoption of secure and reliable DML practices.

For Workers: 1) Transparency and Fairness: The system ensures verifiable task allocation and compensation through tamper-proof smart contracts, promoting trust and eliminating potential biases. 2) Data Protection: ABPRE encryption safeguards worker data from manipulation or privacy violations, empowering them with greater control. 3) Enhanced Reputation: The transparent and secure environment fosters trust, allowing workers to build a strong reputation within the DML ecosystem.

For Requesters: 1) Reliable Workforce: The system facilitates access to a larger pool of motivated workers confident in fair treatment and data security. 2) Reduced Risks: Blockchain technology minimizes the risk of data breaches and malicious behavior by participants. 3) Efficiency and Cost-Effectiveness: The automated and secure nature of the system streamlines workflows and potentially reduces operational costs.

Impact on the DML Ecosystem: 1) Wider Adoption: Enhanced security and trust incentivize broader adoption of DML systems, fostering collaboration and innovation. 2) Market Growth: The secure and reliable environment stimulates growth within the DML market, attracting new participants and investment. 3) Research Advancement: Our system paves the way for further research in secure and decentralized DML solutions, shaping the future of collaborative machine learning.

Our system can be a valuable tool for data sharing between workers and requesters on crowd-sourcing platforms like CrowdWorks, Appen, Labelbox, and Amazon Mechanical Turk, mentioned in Section 1. Additionally, by ensuring that all task-related procedures are transparently recorded on the system, our system enhances trust and can serve as a key motivator for participants to engage in the system.

8.2 Future Work

Our system operates in a task-outsourcing mode, where worker computing resources are utilized. Our approach is effective in mitigating certain security threats. However, different DML modes present unique challenges. For instance, in federated learning mode, workers contribute their training data (models or datasets) to the requester. While the threat of malicious worker remains similar, these workers control the selection of the training dataset. Dataset splitting and sharing are not relevant in this scenario. Therefore, ensuring accountability in other DML modes requires defining and researching new security models specific to each mode. Our future work will focus on extending our research scope to encompass these other modes, thereby ensuring comprehensive accountability across the DML landscape.

Our proposed system leverages smart contracts to ensure a transparent random split of the original dataset into training and test datasets. However, the inherent determinism of Solidity smart contracts raises concerns about the predictability of generated random values, potentially introducing security vulnerabilities. While off-chain solutions like Chainlink Verifiable Random Functions (VRF) [42] offer cryptographically secure randomness, their associated costs necessitate a cost-effective design. To address this trade-off, we plan to implement an off-chain random number generator in a future iteration, achieving a balance between security and cost.

9 Conclusion

This paper proposes a novel and accountable DML system designed to address the challenge of securing training data in environments with potentially malicious participants. Our system leverages the strengths of both IPFS for efficient off-chain storage and blockchain technology for transparent and tamper-proof dataset splitting. We achieve secure and accountable data sharing without relying on trusted third parties through the innovative application of ABPRE. Our contributions are twofold: 1) Verifiable Dataset Splitting: We introduce a blockchain smart contract that executes a verifiable split of training data into training and test sets. This eliminates participant involvement and potential manipulation during the splitting process. 2) Secure and Trustless Collaboration: By leveraging the transparency and immutability of the blockchain, our system enables secure and reliable collaboration even among participants who do not have prior trust in each other. We demonstrate the system's effectiveness through a rigorous security analysis and extensive experiments conducted on the Ethereum and IPFS platforms. These experiments showcase the system's security, practicality, and its potential to revolutionize DML systems. We believe this work paves the way for a future where decentralized systems can become a trusted and reliable platform for collaborative problem-solving.

Acknowledgement: None.

Funding Statement: This research was supported by the MSIT (Ministry of Science and ICT), Korea, under the Special R&D Zone Development Project (R&D)—Development of R&D Innovation Valley support program (2023-DD-RD-0152) supervised by the Innovation Foundation. It was also partially supported by the Ministry of Science and ICT (MSIT), Korea, under the Information Technology Research Center (ITRC) support program (IITP-2024-2020-0-01797) supervised by the Institute for Information & Communications Technology Planning & Evaluation (IITP).

Author Contributions: The authors confirm contribution to the paper as follows: Study conception and design: Siwan Noh, Kyung-Hyune Rhee; data collection: Siwan Noh; analysis and interpretation of results: Siwan Noh; draft manuscript preparation: Siwan Noh, Kyung-Hyune Rhee. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: The data are contained within the article.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] C. Gupta, I. Johri, K. Srinivasan, Y. C. Hu, S. M. Qaisar and K. Y. Huang, “A systematic review on machine learning and deep learning models for electronic information security in mobile networks,” *Sensors*, vol. 22, no. 5, pp. 2017, 2022. doi: 10.3390/s22052017.
[2] A. Hard *et al.*, “Federated learning for mobile keyboard prediction,” arXiv preprint arXiv:1811.03604, 2018.
[3] “Golem Factory,” Accessed: Mar. 22, 2024. [Online]. Available: https://www.golem.network/
[4] “Crowdworks AI,” Accessed: Mar. 22, 2024. [Online]. Available: https://works.crowdworks.kr/
[5] “Appen,” Accessed: Mar. 22, 2024. [Online]. Available: https://www.appen.com/
[6] “Labelbox,” Accessed: Mar. 22, 2024. [Online]. Available: https://labelbox.com/
[7] L. Ouyang, Y. Yuan, and F. Y. Wang, “Learning markets: An AI collaboration framework based on blockchain and smart contracts,” *IEEE Internet Things J.*, vol. 9, no. 16, pp. 14273–14286, 2022. doi: 10.1109/JIOT.2020.3032706.
[8] A. Ali, I. Ilahi, A. Qayyum, I. Mohammed, A. Al-Fuqaha and J. Qadir, “Incentive-driven federated learning and associated security challenges: A systematic review,” 2021. doi: 10.36227/techrxiv.14945433.
[9] M. D. Packard and P. L. Bylund, “From homo economicus to homo agens: Toward a subjective rationality for entrepreneurship,” *J. Bus. Ventur.*, vol. 36, no. 6, pp. 106159, 2021. doi: 10.1016/j.jbusvent.2021.106159.
[10] Y. Fraboni, R. Vidal, and M. Lorenzi, “Free-rider attacks on model aggregation in federated learning,” in *Proc. Int. Conf. Artif. Intell. Stat.*, Apr. 13–15, 2021, pp. 1846–1854.
[11] K. Sarpatwar *et al.*, “Towards enabling trusted artificial intelligence via blockchain,” *Policy-Based Auton. Data Gov.*, pp. 137–153, 2019. doi: 10.1007/978-3-030-17277-0.
[12] H. B. Desai, M. S. Ozdayi, and M. Kantarcioglu, “BlockFLA: Accountable federated learning via hybrid blockchain architecture,” in *Proc. Eleventh ACM Conf. Data Appl. Secur. Priv.*, Apr. 26–28, 2021, pp. 101–112.
[13] S. K. Lo *et al.*, “Toward trustworthy AI: Blockchain-based architecture design for accountability and fairness of federated learning systems,” *IEEE Internet Things J.*, vol. 10, no. 4, pp. 3276–3284, 2022. doi: 10.1109/JIOT.2022.3144450.

[14] C. Ma *et al.*, “When federated learning meets blockchain: A new distributed learning paradigm,” *IEEE Comput. Intell. Mag.*, vol. 17, no. 3, pp. 26–33, 2022. doi: 10.1109/MCI.2022.3180932.
[15] D. C. Nguyen *et al.*, “Federated learning meets blockchain in edge computing: Opportunities and challenges,” *IEEE Internet Things J.*, vol. 8, no. 16, pp. 12806–12825, 2021. doi: 10.1109/JIOT.2021.3072611.
[16] M. M. Salim and J. H. Park, “Federated learning-based secure electronic health record sharing scheme in medical informatics,” *IEEE J. Biomed. Health Inf.*, vol. 27, no. 2, pp. 617–624, 2023. doi: 10.1109/JBHI.2022.3174823.
[17] A. R. Short, H. C. Leligou, M. Papoutsidakis, and E. Theocharis, “Using blockchain technologies to improve security in federated learning systems,” in *Proc. 44th IEEE Annu. Comput., Softw., Appl. Conf. (COMPSAC)*, Madrid, Spain, Jul. 13–17, 2020, pp. 1183–1188.
[18] V. Mugunthan, R. Rahman, and L. Kagal, “BlockFLow: Decentralized, privacy-preserving and accountable federated machine learning,” in *Proc. 3rd Int. Congr. Blockchain Appl.*, Salamanca, Spain, Oct. 6–8, 2021, pp. 233–242.
[19] A. B. Kurtulmus and K. Daniel, “Trustless machine learning contracts; evaluating and exchanging machine learning models on the ethereum blockchain,” arXiv preprint arXiv:1802.10185, 2018.
[20] S. Liu, X. Wang, L. Hui, and W. Wu, “Blockchain-based decentralized federated learning method in edge computing environment,” *Appl. Sci.*, vol. 13, no. 3, pp. 1677–1693, 2023.
[21] “Amazon Web Service,” Accessed: Mar. 22, 2024. [Online]. Available: https://www.mturk.com/
[22] H. Xie, E. Maddalena, R. Qarout, and A. Checco, “The dark side of recruitment in crowdsourcing: Ethics and transparency in micro-task marketplaces,” *Comput. Support. Coop. Work*, vol. 32, no. 3, pp. 439–474, 2023. doi: 10.1007/s10606-023-09464-9.
[23] US Government Accountability Office, “Artificial intelligence: An accountability framework for federal agencies and other entities,” 2021. Accessed: Apr. 29, 2024. [Online]. Available: https://search.proquest.com/docview/2559990674
[24] G. Caldarelli, “Understanding the blockchain oracle problem: A call for action,” *Information*, vol. 11, no. 11, pp. 509, 2020. doi: 10.3390/info11110509.
[25] Q. Li, Z. Wen, and B. He, “Practical federated gradient boosting decision trees,” in *Proc. AAAI Conf. Artif. Intell.*, New York, NY, USA, Feb. 7–12, 2020, vol. 34, pp. 4642–4649. doi: 10.1609/aaai.v34i04.5895.
[26] Q. Wang, Y. Ma, K. Zhao, and Y. Tian, “A comprehensive survey of loss functions in machine learning,” *Ann. Data Sci.*, vol. 9, no. 2, pp. 187–212, 2022. doi: 10.1007/s40745-020-00253-5.
[27] D. Martin, J. O’Neill, N. Gupta, and B. V. Hanrahan, “Turking in a global labour market,” *Comput. Support. Coop. Work*, vol. 25, no. 1, pp. 39–77, 2016. doi: 10.1007/s10606-015-9241-6.
[28] M. S. Silberman, L. Irani, and J. Ross, “Ethics and tactics of professional crowdwork,” *XRDS: Crossroads, ACM Mag. Students*, vol. 17, no. 2, pp. 39–43, 2010.
[29] ChrisTurk, “TurkerViewJS,” 2022. Accessed: Apr. 29, 2024. [Online]. Available: https://turkerview.com/mturk-scripts/1-turkerviewjs
[30] J. Hao, Y. Zhao, and J. Zhang, “Time efficient federated learning with semi-asynchronous communication,” in *Proc. 26th IEEE Int. Conf. Parallel Distrib. Syst. (ICPADS)*, Hong Kong, Dec. 2–4, 2020, pp. 156–163.
[31] A. Imteaj *et al.*, “FedAR: Activity and resource-aware federated learning model for distributed mobile robots,” in *Proc. 19th IEEE Int. Conf. Mach. Learn. Appl. (ICMLA)*, Miami, FL, USA, Dec. 14–17, 2020, pp. 1153–1160.
[32] A. Homeyer *et al.*, “Recommendations on compiling test datasets for evaluating artificial intelligence solutions in pathology,” *Mod. Pathol.*, vol. 35, no. 12, pp. 1759–1769, 2022. doi: 10.1038/s41379-022-01147-y.
[33] P. Rathore, A. Basak, S. H. Nistala, and V. Runkana, “Untargeted, targeted and universal adversarial attacks and defenses on time series,” in *Proc. 2020 Int. Joint Conf. Neural Netw. (IJCNN)*, Glasgow, UK, Jul. 19–24, 2020, pp. 1–8.
[34] Z. Kong *et al.*, “A survey on adversarial attack in the age of artificial intelligence,” *Wirel. Commun. Mob. Comput.*, vol. 2021, pp. 1–22, 2021. doi: 10.1155/2021/4907754.

[35] C. Ge, W. Susilo, J. Baek, Z. Liu, J. Xia and L. Fang, "A verifiable and fair attribute-based proxy re-encryption scheme for data sharing in clouds," *IEEE Trans. Depend. Secur. Comput.*, vol. 19, no. 5, pp. 2907–2919, 2022. doi: 10.1109/TDSC.2021.3076580.

[36] K. Liang *et al.*, "A secure and efficient ciphertext-policy attribute-based proxy re-encryption for cloud data sharing," *Future Gen. Comput. Syst.*, vol. 52, no. 1, pp. 95–108, 2015. doi: 10.1016/j.future.2014.11.016.

[37] K. Liang, L. Fang, W. Susilo, and D. S. Wong, "A ciphertext-policy attribute-based proxy re-encryption with chosen-ciphertext security," in *Proc. 5th Int. Conf. Intell. Netw. Collab. Syst.*, Xi'an, China, Sep. 9–11, 2013, pp. 552–559.

[38] J. Benet, "IPFS—content addressed, versioned, P2P file system," arXiv preprint arXiv:1407.3561, 2014.

[39] P. Maymounkov and D. Mazières, "Kademlia: A peer-to-peer information system based on the XOR metric," in *Proc. Int. Workshop Peer-to-Peer Syst.*, Cambridge, MA, USA, Mar. 7–8, 2002, pp. 53–65.

[40] S. Prusty, S. Patnaik, and S. K. Dash, "SKCV: Stratified k-fold cross-validation on ML classifiers for predicting cervical cancer," *Front. Nanotechnol.*, vol. 4, pp. 054002, 2022. doi: 10.3389/fnano.2022.972421.

[41] E. Wood, "A secure decentralised generalised transaction ledger," *Yellow Paper*, vol. 151, pp. 1–32, 2014.

[42] L. Breidenbach *et al.*, "Chainlink 2.0: Next steps in the evolution of decentralized oracle networks," Chainlink Labs, 2021. Accessed: Apr. 29, 2024. [Online]. Available: https://research.chain.link/whitepaper-v2.pdf