DeRelayL: Sustainable Decentralized Relay Learning

Haihan Duan, *Member, IEEE*, Tengfei Ma, Yuyang Qin, Runhao Zeng, *Member, IEEE*,
Wei Cai, *Senior Member, IEEE*, Victor C. M. Leung, *Life Fellow, IEEE*, and Xiping Hu, *Member, IEEE*

***Abstract*—In the era of Big Data, large-scale machine learning models have revolutionized various fields, driving significant advancements. However, large-scale model training demands high financial and computational resources, which are only affordable by a few technological giants and well-funded institutions. In this case, common users like mobile users, the real creators of valuable data, are often excluded from fully benefiting due to the barriers, while the current methods for accessing large-scale models either limit user ownership or lack sustainability. This growing gap highlights the urgent need for a collaborative model training approach, allowing common users to train and share models. However, existing collaborative model training paradigms, especially federated learning (FL), primarily focus on data privacy and group-based model aggregation. To this end, this paper intends to address this issue by proposing a novel training paradigm named decentralized relay learning (DeRelayL), a sustainable learning system where permissionless participants can contribute to model training in a relay-like manner and share the model. In detail, this paper presents the architecture and workflow of DeRelayL, designs incentive mechanisms to ensure sustainability, and conducts theoretical analysis and numerical simulations to demonstrate its effectiveness.**

***Index Terms*—Relay learning, decentralized model training, sustainable model training, federated learning, blockchain.**

I. Introduction

In THE era of Big Data, the rise of large-scale machine learning models has revolutionized various fields, from natural language processing to computer vision, healthcare, education, and beyond. These models are usually trained on enormous datasets and have demonstrated remarkable capabilities in terms of accuracy, generalization, and predictive ability, which drives significant advancements in technology and scientific discovery [1]. Moreover, the development and deployment of large models are no longer merely trends but necessities for fully capitalizing on the opportunities presented by the Big Data era. Therefore, as the volume of data continues to grow exponentially, advanced machine learning techniques need to harness the information from the available data.

However, a major problem has emerged: although individuals, organizations, and even small enterprises often possess valuable data, completely training large-scale models is obviously beyond the financial and technical ability of common users [2], [3]. Specifically, the computational resources required to train a large-scale model completely can only be afforded by a few technological giants and well-funded institutions [1]. These institutions may purchase training data from third parties, or even directly scrape data from websites without payment, thus the original creators of the knowledge (common users) find it hard to obtain profits. As a result, this forms a growing gap between the giants and common users with insufficient computational resources (e.g., common users using only mobile devices), limiting common users from enjoying the societal benefits of intelligent data-driven insights in the Big Data era created by themselves [4].

In practice, using the application of large language models (LLMs) [1] as an example, common users can actually utilize LLMs based on two approaches provided by the giants. (1) The first way is **close-source**, in which the common users need to pay for the use. The prospective users are usually charged by monthly subscription or accumulation of utilized input/output tokens. In this case, the common users can only obtain the rights to use the models online, but the model weights are not directly accessible to them, i.e., the common users do not truly possess the models. (2) The second approach is **open-source**, where some giants may voluntarily contribute well-trained models for the public to download freely. The common users can really obtain model weights in this situation, and the giants can earn nonmonetary profit, such as reputation. However, the open-source approach is hard to achieve sustainability, since there lacks an explicit monetary incentive to maintain the model update [5]. To this end, this study seeks to address this pressing issue by

Received 13 December 2024; revised 26 February 2025; accepted 2 April 2025. Date of publication 7 April 2025; date of current version 6 August 2025. This work was supported by Guangdong-Hong Kong-Macao Joint Laboratory for Emotional Intelligence and Pervasive Computing, Artificial Intelligence Research Institute, Shenzhen MSU-BIT University. Recommended for acceptance by J. Kang. *(Corresponding author: Xiping Hu.)*

Haihan Duan and Runhao Zeng are with the Artificial Intelligence Research Institute, Shenzhen MSU-BIT University, Shenzhen 518172, China, also with the Guangdong-Hong Kong-Macao Joint Laboratory for Emotion Intelligence and Pervasive Computing, Shenzhen 518172, China (e-mail: duanhaihan@smbu.edu.cn; runhaozeng.cs@gmail.com).

Tengfei Ma and Yuyang Qin are with the Artificial Intelligence Research Institute, Shenzhen MSU-BIT University, Shenzhen 518172, China, also with the Guangdong-Hong Kong-Macao Joint Laboratory for Emotion Intelligence and Pervasive Computing, Shenzhen 518172, China, and also with the The Chinese University of Hong Kong, Shenzhen 518172, China (e-mail: 121090406@link.cuhk.edu.cn; yuyangqin1@link.cuhk.edu.cn).

Wei Cai is with the School of Engineering and Technology, University of Washington, Tacoma, WA 98402-3100 USA (e-mail: weicaics@uw.edu).

Victor C. M. Leung is with the Artificial Intelligence Research Institute, Shenzhen MSU-BIT University, Shenzhen 518172, China, and also with the Department of Electrical and Computer Engineering, University of British Columbia, Vancouve, BC V6T 1Z4, China (e-mail: vleung@ieee.org).

Xiping Hu is with the Artificial Intelligence Research Institute, Shenzhen MSU-BIT University, Shenzhen 518172, China, also with the Guangdong-Hong Kong-Macao Joint Laboratory for Emotion Intelligence and Pervasive Computing, Shenzhen 518172, China, and also with the School of Medical Technology, Beijing Institute of Technology, Beijing 100090, China (e-mail: huxp@bit.edu.cn).

This article has supplementary downloadable material available at https://doi.org/10.1109/TMC.2025.355854, provided by the authors.

Digital Object Identifier 10.1109/TMC.2025.3558544

proposing a sustainable decentralized learning system in which participants can train like a relay and collaboratively share the model, named Decentralized Relay Learning (DeRelayL), i.e., participants who have sufficient contributions to the relay-like model training can possess the trained model weights, acting like **semi-open-source**.

In recent years, some researchers have discussed the collaborative model training methodologies. Among them, federated learning (FL) is the most notable framework for collaborative model training with privacy preservation [6], [7]. Traditional FL relies on centralized model parameter aggregation and faces challenges like performance degradation with non-IID data [6], [8], [9]. Moreover, other researchers also investigate decentralized FL [10], as well as blockchain-enabled FL [11], which mainly studies decentralized model parameter aggregation [12], [13] and decentralization-related topics [14], [15], [16], [17], [18], [19], [20], [21], [22], [23]. However, the core motivation of FL differs significantly from the proposed DeRelayL, where FL typically revolves around the challenges of the group-based aggregation process, but DeRelayL focuses more on motivating independent participants to sustainably contribute to and benefit from model training. Besides the FL, other existing studies also have investigated collaborative model training, discussing collaborations in volunteer computing environments [24], [25], secure multi-party collaborative model training [26], [27], decentralized LLM training [28], etc. The most relevant study named relay learning presented by Bo et al. [29], but they focused on security and privacy issues in relay-like model training between clinical multi-sites, so the motivation and application are quite different from our study.

In this paper, we aim to build a sustainable decentralized learning system based on blockchain, where permissionless participants can collaboratively train and share models. The models will be passed among participants in the learning system, following a relay-like learning process. In each round, the model evolves based on the previous round's updates, creating a continuous chain of collaborative learning. Only the participants who have contributed to the model training or maintaining the system operation can share the models, which could ensure that each participant's contribution is recognized and rewarded. Supported by blockchain, this procedure fosters a decentralized and collaborative learning environment, where different trainers can take over the process at different stages, allowing for a more flexible and efficient model development. Therefore, the paradigm operates like a relay, where the task is passed from one participant to another in sequence, with each participant contributing their part in a coordinated manner, so-called **Decentralized Relay Learning (DeRelayL)**.

The contributions of this paper can be concluded as follows:

- This paper proposes a novel learning paradigm regarding the sustainable decentralized collaborative model training. Specifically, we introduce the architecture of DeRelayL based on blockchain and present a detailed system workflow. To the best of our knowledge, there are few existing studies that share the same considerations.
- To maintain the sustainability of DeRelayL, we formulate the utilities of all participants and design a corresponding incentive mechanism to guarantee the participants' Individual Rationality (IR) and Incentive Compatibility (IC).
- This paper conducts a detailed theoretical analysis to formulate a condition set for the incentive mechanism. Moreover, we also design a numerical simulation to demonstrate the effectiveness of the incentive mechanism and the sustainability of DeRelayL.
- Due to the complexity of DeRelayL, this paper can only present the key motivation and system workflow, while some techniques in realistic implementation are not mature enough. To this end, we also comprehensively discuss the potential challenges and research directions of DeRelayL.

II. Related Work

In this section, we mainly discuss existing collaborative model training paradigms, including federated learning (FL) and other related collaborative training methods, to clarify the different motivations and scenarios between the existing methodologies and the proposed DeRelayL.

A. Federated Learning

Federated learning (FL) is a distributed machine learning approach that enables multiple nodes to collaboratively train a shared model while keeping local data private [6]. Generally, FL assumes that participants exchange model updates with a central server that aggregates them, instead of sharing raw data. Therefore, the typical FL algorithm studies the aggregation of gradients, such as FedAvg [6]. On the other hand, some studies point out that FL faces the challenge of performance degradation in non-IID data [8], [9]. Many solutions have been proposed to solve the problem, including optimizing the model aggregation [30], [31], knowledge distillation [32], regularizing training in distributed nodes [33], and Bayesian reformulation [34]. Besides the basics of FL, some researchers also study related topics, such as balancing personalization and generalization [35], acceleration [36], [37], privacy and fairness preserving [38], [39].

On the other hand, some researchers have also noticed that traditional FL relies on a centralized server for the aggregation of model parameters or gradients, so decentralized FL has been studied in recent years [10]. The decentralized FL mainly focuses on distributed model parameter aggregation between the neighboring participants [12], [13]. Referring to decentralization, blockchain is the cutting-edge implementation of decentralized systems, and many researchers have paid attention to blockchain-enabled FL [11], studying the architecture of decentralized FL [14], consensus algorithm [15], [16], [40], decentralized aggregator assignment [17], [18], resource trading and allocation [19], [20], defending against poisoning attacks [21], [22], incentive mechanism design [23], etc.

However, the motivation of FL shows a significant difference from that of the proposed DeRelayL. As shown in Fig. 1, we demonstrate a comparative diagram with the simplest case to show the difference between FL and DeRelayL. FL mainly discusses the collaborative model training among a group, typically involving an aggregation process per round, but DeRelayL emphasizes incentivizing a sustainable collaborative model

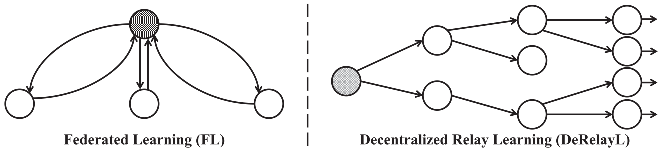

Fig. 1. A comparative diagram between federated learning and relay learning.

training, which is expected to present like a relay among multiple independent participants, who can obtain the model if they have contributions to the DeRelayL system. Although Buyukates et al. [41] presented similar considerations, where they proposed a proof-of-contribution-based design for collaborative machine learning on the blockchain, the trained model still belongs to the initiator rather than the participants. Similarly, subsequent studies of contribution proof can refer to Ebrahimi et al. [42] and Yazdaninejad et al. [43].

B. Other Collaborative Model Training

Although FL is the best-known paradigm of collaborative model training, some other methodologies present different considerations. Diskin et al. [24] proposed a framework for distributed deep learning in open collaborations (DeDLOC), addressing the challenges posed by volunteer computing environments. Ryabinin et al. [25] introduced a decentralized mixture-of-experts (DMoE) model designed to leverage volunteer computing for training large neural networks, especially in distributing the computational workload across unreliable hardware. Zheng et al. [26] presented Cerebro, a platform designed for secure multi-party cryptographic collaborative learning, avoiding exposing sensitive information when combining data from multiple organizations. The most similar motivation was mentioned by Gao et al. [28], who proposed a theoretical design of a decentralized LLM and used GradientCoin to incentivize model training, but it lacks a verification of the training, so unreliable participants will tend to cheat for incentive without real training. The most relevant study was proposed by Bo et al. [29], in which the authors also applied the term relay learning to present their paradigm. However, they mainly considered the security and privacy issues in relay-like model training between clinical multi-sites (one by one), so the motivation and application are quite different from our study.

Overall, few existing studies about collaborative model training have shown the same consideration as our proposal, which intends to build a decentralized collaborative model training system that can sustainably work.

III. System Design

This section will present the system design. First, we will discuss the motivation and challenges regarding the proposed blockchain-based DeRelayL system, which also reflects our core considerations during the architecture design. Then, we will illustrate the system architecture, addressing the aforementioned challenges. At last, the corresponding mechanism design and problem formulation will be investigated.

A. Motivation and Challenges

As discussed in Section II (also refer to Fig. 1), the DeRelayL shows significant differences compared with FL [44]. In conclusion, the motivation of DeRelayL is **to build a sustainable decentralized learning system in which participants can train like a relay and collaboratively share the model**. To this end, this subsection will first clarify the challenges following with the logic of the proposed motivation.

C1. Sustainability of the training system: Generally, the participants of the training system will keep the strategies that can maximize their utilities. Under this assumption, the open-source model is unsustainable, e.g., if everyone can obtain the model without cost, nobody will have the motivation to train the model, since the training process has cost, which will decrease the participant's utility. Therefore, the system design needs to guarantee that the model can be obtained only if the participants have contributed to the whole procedure of DeRelayL, e.g., participating in the model training or supporting the blockchain system operation.

C2. Model weight leakage before the model training: Assuming that a participant wants to contribute to the system by training models, the most common and efficient way is to ask the model owner to send the model to the participant. This step faces a risk that the participant may not fulfill the training duty after obtaining the model, especially in a decentralized system, which means that the participant can obtain a model without any cost. To address the challenge, there should be punishment for the dishonest participants, so the prospective trainers should deposit a certain cost before obtaining the model (the deposit cost should be higher than the model's value), which will be returned after honest behavior. In an extreme situation, if the participants cannot finish the model training subjectively/objectively, the process is equivalent to a transaction between the participants and the system, where the participants spend the deposit cost to buy the model.

C3. Dishonest model owners that provide fake models to the participants: As we discussed in **C2**, the participants need to deposit a certain cost for requesting the model and withdraw the cost after training. However, dishonest model owners may provide fake models to the participants or even do not respond. In this case, the prospective trainers will lose the model and the deposit at the same time. Therefore, the model owner should also deposit a certain cost by constructing a smart contract with the prospective trainers, which will be returned at the same time as the deposit from the trainer is returned. More importantly, under this setting, the model owner and the trainer form a community of interests, so there should be a two-way selection mechanism between model owners and prospective trainers for them to find a reliable partner.

C4. Evaluation of model training: In a decentralized system, it is difficult for model trainers to prove the completed training process. For instance, dishonest trainers can add some white noise to the model to pass the check of model hash, claiming that they finished the model training. Therefore, it requires an evaluation to validate the training, which should be provided by a random third party. Then, a new challenge appears, how to

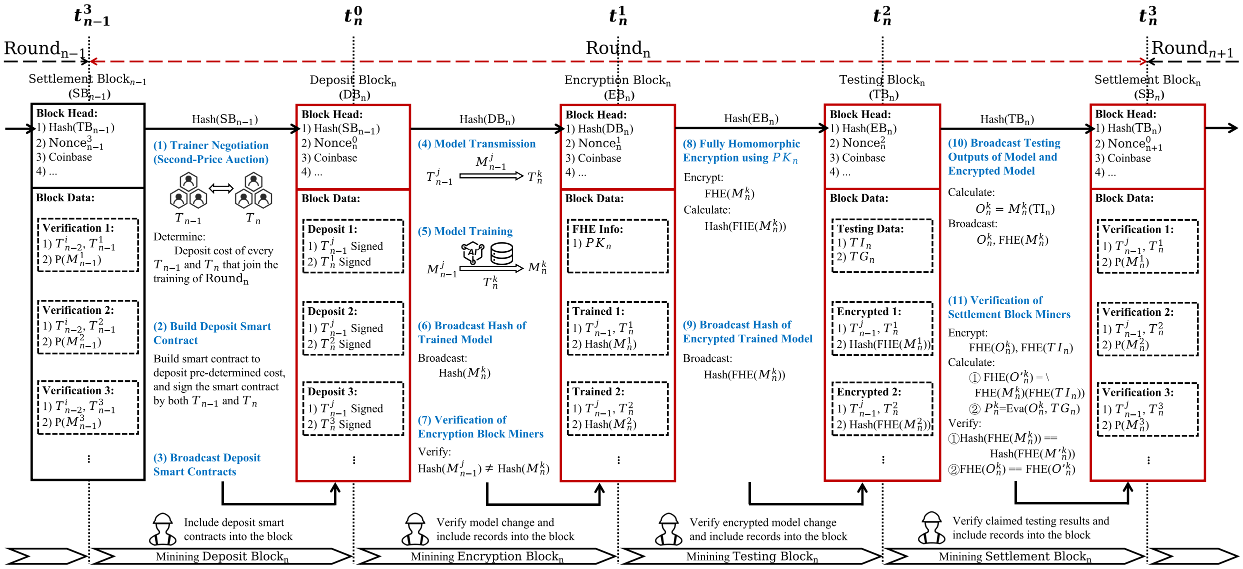
t_{n-1}^{3}
t_{n}^{0}
t_{n}^{1}
t_{n}^{2}
t_{n}^{3}
Round_{n-1}
Round_{n}
Round_{n+1}
Settlement Block_{n-1} (SB_{n-1})
Block Head:
1) $\mathrm{Hash}(\mathrm{TB}_{n-1})$
2) Nonce_{n-1}^{3}
3) Coinbase
4) ...
Block Data:
Verification 1:
1) T_{n-2}^{i}, T_{n-1}^{1}
2) $\mathrm{P}(M_{n-1}^{1})$
Verification 2:
1) T_{n-2}^{i}, T_{n-1}^{2}
2) $\mathrm{P}(M_{n-1}^{2})$
Verification 3:
1) T_{n-2}^{i}, T_{n-1}^{3}
2) $\mathrm{P}(M_{n-1}^{3})$
$\mathrm{Hash}(\mathrm{SB}_{n-1})$
(1) Trainer Negotiation (Second-Price Auction)
T_{n-1}
T_{n}
Determine:
Deposit cost of every T_{n-1} and T_{n} that join the training of Round_{n}
(2) Build Deposit Smart Contract
Build smart contract to deposit pre-determined cost, and sign the smart contract by both T_{n-1} and T_{n}
(3) Broadcast Deposit Smart Contracts
Deposit Block_{n} (DB_{n})
Block Head:
1) $\mathrm{Hash}(\mathrm{SB}_{n-1})$
2) Nonce_{n}^{0}
3) Coinbase
4) ...
Block Data:
Deposit 1:
1) T_{n-1}^{j} Signed
2) T_{n}^{1} Signed
Deposit 2:
1) T_{n-1}^{j} Signed
2) T_{n}^{2} Signed
Deposit 3:
1) T_{n-1}^{j} Signed
2) T_{n}^{3} Signed
$\mathrm{Hash}(\mathrm{DB}_{n})$
(4) Model Transmission
$T_{n-1}^{j} \xrightarrow{M_{n-1}^{j}} T_{n}^{k}$
(5) Model Training
AI
$M_{n-1}^{j} \xrightarrow[T_{n}^{k}]{} M_{n}^{k}$
(6) Broadcast Hash of Trained Model
Broadcast:
$\mathrm{Hash}(M_{n}^{k})$
(7) Verification of Encryption Block Miners
Verify:
$\mathrm{Hash}(M_{n-1}^{j}) \neq \mathrm{Hash}(M_{n}^{k})$
Encryption Block_{n} (EB_{n})
Block Head:
1) $\mathrm{Hash}(\mathrm{DB}_{n})$
2) Nonce_{n}^{1}
3) Coinbase
4) ...
Block Data:
FHE Info:
1) PK_{n}
Trained 1:
1) T_{n-1}^{j}, T_{n}^{1}
2) $\mathrm{Hash}(M_{n}^{1})$
Trained 2:
1) T_{n-1}^{j}, T_{n}^{2}
2) $\mathrm{Hash}(M_{n}^{2})$
$\mathrm{Hash}(\mathrm{EB}_{n})$
(8) Fully Homomorphic Encryption using PK_{n}
Encrypt:
$\mathrm{FHE}(M_{n}^{k})$
Calculate:
$\mathrm{Hash}(\mathrm{FHE}(M_{n}^{k}))$
(9) Broadcast Hash of Encrypted Trained Model
Broadcast:
$\mathrm{Hash}(\mathrm{FHE}(M_{n}^{k}))$
Testing Block_{n} (TB_{n})
Block Head:
1) $\mathrm{Hash}(\mathrm{EB}_{n})$
2) Nonce_{n}^{2}
3) Coinbase
4) ...
Block Data:
Testing Data:
1) TI_{n}
2) TG_{n}
Encrypted 1:
1) T_{n-1}^{j}, T_{n}^{1}
2) $\mathrm{Hash}(\mathrm{FHE}(M_{n}^{1}))$
Encrypted 2:
1) T_{n-1}^{j}, T_{n}^{2}
2) $\mathrm{Hash}(\mathrm{FHE}(M_{n}^{2}))$
$\mathrm{Hash}(\mathrm{TB}_{n})$
(10) Broadcast Testing Outputs of Model and Encrypted Model
Calculate:
$O_{n}^{k} = M_{n}^{k}(\mathrm{TI}_{n})$
Broadcast:
$O_{n}^{k}, \mathrm{FHE}(M_{n}^{k})$
(11) Verification of Settlement Block Miners
Encrypt:
$\mathrm{FHE}(O_{n}^{k}), \mathrm{FHE}(TI_{n})$
Calculate:
① $\mathrm{FHE}(O'^{k}_{n}) = \backslash$ $\mathrm{FHE}(M_{n}^{k})(\mathrm{FHE}(TI_{n}))$
② $P_{n}^{k}=\mathrm{Eva}(O_{n}^{k}, TG_{n})$
Verify:
①$\mathrm{Hash}(\mathrm{FHE}(M_{n}^{k})) == \mathrm{Hash}(\mathrm{FHE}(M'^{k}_{n}))$
②$\mathrm{FHE}(O_{n}^{k}) == \mathrm{FHE}(O'^{k}_{n})$
Settlement Block_{n} (SB_{n})
Block Head:
1) $\mathrm{Hash}(\mathrm{TB}_{n})$
2) Nonce_{n+1}^{0}
3) Coinbase
4) ...
Block Data:
Verification 1:
1) T_{n-1}^{j}, T_{n}^{1}
2) $\mathrm{P}(M_{n}^{1})$
Verification 2:
1) T_{n-1}^{j}, T_{n}^{2}
2) $\mathrm{P}(M_{n}^{2})$
Verification 3:
1) T_{n-1}^{j}, T_{n}^{3}
2) $\mathrm{P}(M_{n}^{3})$
Include deposit smart contracts into the block
Verify model change and include records into the block
Verify encrypted model change and include records into the block
Verify claimed testing results and include records into the block
Mininig Deposit Block_{n}
Mininig Encryption Block_{n}
Mininig Testing Block_{n}
Mininig Settlement Block_{n}

Fig. 2. System architecture and workflow of sustainable decentralized relay learning (DeRelayL).

determine whether the trainer has honestly finished the training. Moreover, the black-box training of neural networks and different data distributions between trainers and the evaluation data providers will also influence the performance evaluation, e.g., it is hard to avoid that a dishonest but lucky trainer obtained the highest performance by only adding white noise. To this end, a relatively fair evaluation method is necessary.

C5. Model weight leakage during the performance evaluation: To evaluate the performance, it requires the output from the trained model by inputting testing data. Obviously, it is unreliable that the trainers test their models by themselves, while the evaluation of a third party will face the risk of model weight leakage during the transmission. To address the challenge, this paper considers applying fully homomorphic encryption (FHE) [45] to transmit model weights, where FHE is an encryption scheme that enables functions to be run directly on encrypted data while yielding the same encrypted results as if the functions were run on plaintext [46]. With FHE, it is not necessary to calculate testing output using the original trained model, avoiding the model weight leakage.

C6. Verification of the performance evaluation: Following **C5**, the trained model after FHE can be broadcast in the system, so the encrypted model can be obtained by every user. This means that, if the testing data and public key of FHE are publicly available, every user can verify the claimed performance, according to the easy-to-check principle in a decentralized system. This public verification from other users is the fundamental guarantee of a valid training record.

C7. Re-training the model after testing data publication: The performance evaluation process contains a hidden pre-condition that the testing data should be published before the evaluation. Therefore, it is possible that trainers re-train their models based on training data to pass the performance evaluation. To solve the problem, the system should have a mechanism to guarantee the models in the performance evaluation are trained before publishing the testing data.

C8. Collusion between testing data publisher, performance verifier, and trainer: Decentralized systems have a common challenge that some participants may collude with each other. The folking of blockchain can naturally address the challenge since other honest participants who find out the dishonest behavior will spontaneously follow the honest blocks. Globally, the collusion can only obtain short-term benefits, while, at a long-term level, honest participants will share more powerful models with the increase of the blockchain. Therefore, rational participants will behave normally to seek long-term benefits.

B. System Architecture and Workflow

After discussing the design motivation and challenges in Section III-A, this subsection will introduce the system architecture of sustainable Decentralized Relay Learning (DeRelayL), as shown in Fig. 2, by discussing the blockchain design, user roles, and workflow.

The DeRelayL system is based on blockchain, where we define four kinds of blocks according to the different functions during different stages, including deposit block, encryption block, testing block, and settlement block. **1) Deposit Block (DB):** to store deposit smart contracts, which are the fundamental guarantee of model transmission; **2) Encryption Block (EB):** to publish information about FHE and record the hash value of trained models; **3) Testing Block (TB):** to publish testing data and record the hash value of trained models encrypted by the public key in **Encryption Block** using FHE; **4) Settlement Block (SB):** to verify and store the performance of trained models. Note that, the mining process is identical for all blocks, where the core difference is that the miners should act in different roles and include different data corresponding to the stage. By default, we utilize the Proof of Work (PoW) [47] consensus model for block generation as an example, while other consensus models are also available. The detailed usage of each block will be introduced after the discussion of the workflow. In Fig. 2, we illustrate the SB_{n-1} in $Round_{n-1}$ and all blocks in $Round_n$. The whole workflow of the DeRelayL system has 11 main steps:

(1) Trainer Negotiation (Second-Price Auction): As we mentioned, SB records the performance of trained models, as well as their corresponding trainers and resources (details of SB will be discussed in **Step (11)**). Therefore, after the confirmation of SB_{n-1}, all nodes in the system can find the models' performance of the (n-1)$-th$ training round. Abstractly, all participants can be denoted as trainers T, where T_{n-1} are trainers in $Round_{n-1}$ and also the model owners in $Round_n$, and T_n are trainers who will negotiate with model owners T_{n-1} to obtain a model for training in $Round_n$. This negotiation will determine that the trainers T_n will follow which model owner T_{n-1} to train the model, which is a two-way selection procedure. Moreover, each of T_{n-1} and T_n should also determine the deposit cost to participate in $Round_n$. In this paper, we simply apply the second-price auction [48] to complete the procedure (details will be discussed in Section III-C1), while other two-way selection methods may also fit this scenario. Note that this paper assumes that the single trainer can only complete the training of one model in each round, but model owners can select multiple trainers if they have enough coins to deposit (also known as total deposit budget $\mathcal{B}$ in Section III-C1).

(2) Build Deposit Smart Contract: After the two-way selection, T_{n-1} and T_n who completed the procedure should build a smart contract to deposit the predetermined cost. The smart contract should be signed by both T_{n-1} and T_n. The deposit will be returned if the trained model shows good performance in the testing data, but it also faces a risk of loss if the trainers cannot perform normally or even do not finish the training (details can refer to **Step (11)**). Therefore, the two-way selection in **Step (1)** is necessary, because both T_{n-1} and T_n hope to find a reliable partner to minimize risk.

(3) Broadcast Deposit Smart Contracts: The deposit smart contracts will be broadcast after the double signature. DB miners will include the smart contracts and construct DB_n at t_n^0 when they find $Nounce_n^0$. The DB block contains a coinbase transaction to earn a mining reward, which depends on the included deposit smart contracts, prompting miners to pack records as much as possible. This setting is similar to gas fees on a public blockchain (e.g., BitCoin [47]), while the incentive is from the mechanism rather than the users.

(4) Model Transmission: After the publishing of DB at t_n^0, the system enters the training period, where trainers T_n will receive the model from T_{n-1}. In Fig. 2, we illustrate an example that T_n^k will receive a model M_{n-1}^j from T_{n-1}^j.

(5) Model Training: After the model transmission, the trainer T_n^k will train the model using the computational resource and data, where we use M_n^k to denote the trained model. In an unreliable decentralized system, some trainers may fail to finish the model training due to unknown reasons. Correspondingly, as a punishment, the deposit of both T_{n-1} and T_n will not be returned.

(6) Broadcast Hash of Trained Model: After the model training, trainers will broadcast the hash value of trained models to claim that they completed the training process, denoted as $Hash(M_n^k)$.

(7) Verification of Encryption Block Miners: In this step, EB miners will check the hash value of the trained model, verifying $Hash(M_{n-1}^j) \neq Hash(M_n^k)$, which means the updated model is at least different from the original one. Thanks to the transparency of blockchain, the required information can be easily obtained from the previous blocks, e.g., obtain T_{n-1}^j from DB_n according to T_n^k, and then obtain $Hash(M_{n-1}^j)$ from EB_{n-1} based on T_{n-1}^j. When finding $Nounce_n^1$ at t_n^1, the EB miners will include all records that passed the hash check, containing metadata (T_{n-1}^j and T_n^k) and $Hash(M_n^k)$. More importantly, there is a necessary step that EB miners need to calculate a public key PK_n for FHE. EB miners are special compared with other block miners, because they can obtain trained models by their private key of FHE. On the one hand, due to the cost of generating the FHE key pairs, malicious miners may upload a random number as PK_n, influencing the subsequent steps of the system. On the other hand, the model is provided as a reward, incentivizing miners to actively participate and preventing EB miners to access valuable models and exit the system without further contribution. Therefore, obtaining a trained model can guarantee incentive compatibility (IC) for EB miners and motivate them to behave honestly. As long as EB miner does not disrupt the process and their actions contribute positively to the overall integrity of the system, the system remains sustainable.

(8) Fully Homomorphic Encryption using PK_n: After the publishing of EB at t_n^1, all participants of the system can obtain PK_n. Then, the trainers can encrypt their trained model using FHE by PK_n, denoted as $FHE(M_n^k)$, and the corresponding hash value can be formulated as $Hash(FHE(M_n^k))$. This step can fix the trained models before publishing the testing data, ensuring the trainers cannot re-train the models.

(9) Broadcast Hash of Encrypted Trained Model: The trainers will broadcast $Hash(FHE(M_n^k))$ after model encryption. The TB miners will include the hash values and construct TB_n at t_n^2 when they find $Nounce_n^2$. Besides the hash values, the TB miners should also provide testing data to evaluate the performance of trained models, including testing input TI_n and testing ground truth TG_n (if the testing data is too large to store in the blockchain, the TB miners can also provide a decentralized storage address of the data). The testing data are publicly available, which corresponds to three advantages: 1) all participants can verify the results of performance evaluation, ensuring the procedure is valid with consensus of most participants; 2) the quality of the testing data can be supervised by all participants, and other participants can choose to folk the training blockchain if the quality is unsatisfactory; 3) the testing data can be utilized as training data in the next round, globally contributing additional information to the whole DeRelayL system.

(10) Broadcast Testing Outputs of Model and Encrypted Model: After the publishing of TB at t_n^2, all trainers are accessible to the testing data TI_n and TG_n. The trainers can calculate the testing outputs of their models as $O_n^k = M_n^k(TI_n)$. Then, the trainers will broadcast the outputs O_n^k and the encrypted models $FHE(M_n^k)$ (discussed in **Step (8)**) for performance evaluation.

(11) Verification of Settlement Block Miners: The last step is most critical, which involves four parts: model performance

verification, packing valid training records, returning qualified deposits, and additional citation reward. **1) Model performance verification:** The SB miners should verify whether the received outputs O_n^k are generated from M_n^k based on FHE, without obtaining the original model M_n^k. The SB miners will encrypt received outputs O_n^k and testing input TI_n from TB_n using the PK_n from EB_n, denoted as $FHE(O_n^k)$ and $FHE(TI_n)$. Then, SB miners will calculate new outputs using received encrypted models $FHE(M_n^k)$ and encrypted testing input $FHE(TI_n)$, denoted as $FHE(O_n'^k) = FHE(M_n^k)(FHE(TI_n))$. Moreover, the SB miners will also calculate a quantitative performance index P_n^k based on predefined evaluation metrics (e.g., accuracy, mean-square error (MSE), precision), using the received outputs O_n^k. After that, the SB miners will firstly verify whether the received encrypted model is the one confirmed in TB_n, i.e., $Hash(FHE(M_n^k))$ should be equal to received $Hash(FHE(M_n'^k))$. Secondly, the SB miners will verify whether the received outputs are calculated from the claimed model, i.e., $FHE(O_n'^k)$ should be equal to $FHE(O_n^k)$ according to the features of FHE [45]. Note that, due to the transparency, the verification can be checked by any other participant, ensuring the validity. **2) Packing valid training records:** After the verification, SB miners will include valid training records into the block SB_n, including the original model owners T_{n-1}^j, current trainers T_n^k, and corresponding model performance indexes P_n^k. **3) Returning qualified deposit:** Generally, the verification records in SB_n means that the trainer T_n^k has finished the training process, and the smart contracts built in **Step (2)** will return the deposit to both T_{n-1}^j and T_n^k. However, as discussed in **C4** of Section III-A, lazy workers can add very subtle white noise into the original model, which will not significantly influence the model performance. In this case, the updated model can also pass the verification, which means lazy workers can obtain a model at nearly free cost. To address the problem, we set a threshold to increase the risks of participating in the training system, where only the trained models which has performance ranking in top-$\mathcal{K}$ can return the deposit cost for both original model owners T_{n-1}^j and trainers T_n^k. However, due to some uncertain factors, such as testing data distribution and randomness in model training, the top-$\mathcal{K}$ ranking mechanism cannot completely filter lazy workers, e.g., the model from a very lucky lazy worker may achieve better performance than others. The aforementioned case is very special with a low possibility, because a fundamental assumption of the system is that the model performance will generally increase with the honest model training behavior. In fact, the introduction of the top-$\mathcal{K}$ ranking mechanism will also change the trainer negotiation in **Step (1)**, in which both original model owners T_{n-1}^j and trainers T_n^k will be more serious when evaluating and selecting their partners. **4) Additional citation reward:** Besides returning the deposit, we also design a mechanism to reward the original model owners T_{n-1}^j of the top-$\mathcal{K}$ models in $Round_n$, named citation reward. Note that, the original model owners T_{n-1}^j may not be the top-$\mathcal{K}$ models in $Round_{n-1}$. Therefore, although there might be some very lucky lazy workers who occupied the top-$\mathcal{K}$ positions in $Round_{n-1}$, the citation reward can motivate honest trainers at a long-term level, since the trainers T_n^k may not choose the model to follow only based on the ranking of $Round_{n-1}$, while other information of the original model owners T_{n-1}^j (e.g., historical rankings, frequency of participation) will also be considered.

Algorithm 1: Trainer Selection Based on Deposit Bids.

```
**Input:** $\mathbb{T} = \{(T_1, b^{T_1}), (T_2, b^{T_2}), \ldots, (T_{Q_T}, b^{T_{Q_T}})\}$ as the set of trainers and their corresponding deposit bids $b^{T_i}$, MO's deposit cost for one trainer $b^{MO}$, and MO's total deposit budget $\mathcal{B}$.
**Output:** Selected Trainers and their deposit amounts.
$Q_{\text{Selected}} \leftarrow \lfloor \frac{\mathcal{B}}{b^{MO}} \rfloor$
$\mathbb{T}_{\text{Sorted}} = Sort(\mathbb{T}, 2)$    ▷ Sorted by bids in descending
**for** $i = 1$ to $Q_{\text{Selected}} - 1$ **do**
    $\mathbb{T}_{\text{Selected}}[i] \leftarrow \mathbb{T}_{\text{Sorted}}[i][1]$    ▷ Trainers
    $\mathbb{D}_{\text{Deposit}}[i] \leftarrow \mathbb{T}_{\text{Sorted}}[i+1][2]$    ▷ Deposits
**end for**
$\mathbb{T}_{\text{Selected}}[Q_{\text{Selected}}] \leftarrow \mathbb{T}_{\text{Sorted}}[Q_{\text{Selected}}][1]$    ▷ Trainers
$\mathbb{D}_{\text{Deposit}}[Q_{\text{Selected}}] \leftarrow \mathbb{T}_{\text{Sorted}}[Q_{\text{Selected}}][2]$    ▷ Deposits
**Return** $\mathbb{T}_{\text{Selected}}, \mathbb{D}_{\text{Deposit}}$
```

With the increase of training rounds, the above 11 steps will repeat until the model ability converges to an ultimate level without sufficient performance increment due to the limitation of the model size, referring to scaling law [2], [3].

C. Formulation and Mechanism Design

In Section III-B, we utilize T_{n-1} and T_n to denote original model owners and trainers for a general understanding of the cyclic system. In the following parts, we will apply abbreviations of each role to better explain the formulation, i.e., model owner (MO), trainer (T), deposit block miner (DBM), encryption block miner (EBM), testing block miner (TBM), and settlement block miner (SBM).

1) Trainer Negotiation Algorithm: In **Step (1)** of Section III-B, the trainers will negotiate with the original model owners to obtain an opportunity to join the model training, which is a two-way selection that also determines the deposit cost of the model owners and trainers. The two-way selection can be very complex by considering many factors such as historical ranking and participation frequency, but, to simplify the mechanism modeling in this paper, we design a second-price auction [48] that greedily selects model owners with higher ranking and trainers with higher deposit willingness. At first, the model owners will broadcast their pre-determined deposit cost b^{MO} to the trainers. The prospective trainers (totally Q_T trainers) will send sealed messages to model owners to honestly provide their reserve deposit bids b^{T_i}. After that, the model owners can rank the prospective trainers based on the deposit bids. Then, a model owner can greedily select prospective trainers following Algorithm 1 constrained by the total deposit budget $\mathcal{B}$ (to select total $\lfloor \frac{\mathcal{B}}{b^{MO}} \rfloor$ trainers). The model owners will invite the selected trainers to build deposit smart contracts by depositing the second price, and, correspondingly, the prospective trainers will greedily accept invitations from higher-ranking model owners.

TABLE I
KEY ANNOTATIONS (IN THE ORDER OF APPEARANCE)

Symbol	Description	Symbol	Description
$U^{\text{Participant}}$	Utility of Participant, $Participant \in \{MO, T, DBM, EBM, TBM, SBM\}$	$R^{\text{Participant}}$	Revenue of Participant, $Participant \in \{MO, T, DBM, EBM, TBM, SBM\}$
$C^{\text{Participant}}$	Cost of Participant, $Participant \in \{MO, T, DBM, EBM, TBM, SBM\}$	R^{MO}_{Now}	Immediate revenue of MO available now
R^{MO}_{Future}	Revenue of MO available in the future	C^{MO}_{Deposit}	Cost incurred by MO due to potential deposit loss risk
C^{MO}_{Transmit}	Cost incurred by MO for transmitting model parameters	$\overline{Q^{MO}_{\text{Selected}}}$	Average number of Trainers selected by MO
$\mathcal{R}_{\text{Cited}}$	Citation reward for MO when its model is successfully trained (cited) by one Trainer, noticing all $\mathcal{R}$ adjustable by the system	β	Discount rate for T or MO on $\mathcal{R}_{\text{Cited}}$, used to estimate future potential rewards
Q_{Selected}	Number of Deposits signed by an MO in a given round	s	Selection rate of trained models, $0 < s < 1$, e.g., $s = 0.5$ means half of the trained models are selected
b^{MO}	MO's bid: number of coins deposited by MO in the Deposit smart contract for each T	k_{Transmit}	Transmission coefficient, multiplied by the number of model parameters, representing the cost of receiving or transmitting the model
$\|M\|$	Number of model parameters	U^{MO}_{N}	Utility of MO under the strategy "Normal"
$C^{MO}_{\text{DepositLoss}}$	Cost to MO due to deposit loss from intentionally failing to transmit model parameters	U^{MO}_{NTm}	Utility of MO under the strategy "Not Transmitting" model parameters
R^{T}_{Now}	Immediate revenue of T available now	R^{T}_{Future}	Revenue of T available in the future
C_{Train}	Cost of training a model	C^{T}_{Deposit}	Cost incurred by T due to potential deposit loss risk
C^{T}_{RecM}	Cost incurred by T when receiving model parameters from MO	C_{Encrypt}	Cost of encrypting parameters using public key of FHE from EBM
$C_{\text{Broadcast}}$	Cost of broadcasting the FHE-encrypted model parameters	R_{RecM}	The model received by T itself, which could serve as a reward in the formulation
R_{TrainedM}	The model trained by T itself, which could serve as a reward in the formulation	R^{T}_{Cited}	Citation reward for T when its model is successfully trained (cited) by future Ts
V_{RecM}	Version of received model	V^{T}_{Now}	Version of the latest model owned by T just now before receiving new model
©	One unit of reward coin	$\overline{Q^{T}_{\text{Selected}}}$	Average number of selected future Ts who will train model based on the T directly or indirectly
P_{Comp}	Price of computation per unit of time and per unit of data volume	D	Data volume
τ	Training time	b^{T}	T's bid (the number of coins deposited by T)
k_{Encrypt}	Cost of encrypting per unit of model parameters	$Q_{\text{Broadcast}}$	Number (Quantity) of recipients that T needs to broadcast the encrypted model parameters to
k_{Expand}	Factor by which FHE encryption expands parameter size	U^{T}_{N}	Utility of T under the strategy "Normal"
$C^{T}_{\text{DepositLost}}$	Deposit lost by T due to poor behaviors	U^{T}_{NTr}	Utility of T under the strategy "Not Training"
U^{T}_{NBr}	Utility of T under the strategy "Not Broadcasting"	R^{DBM}_{Include}	Revenue for DBM from including deposit information into the blockchain
C_{Mine}	Cost of mining under the PoW consensus mechanism	Q_{Deposit}	Quantity of deposits included on-chain by one DBM
$\mathcal{R}_{\text{Deposit}}$	Revenue per deposit included on-chain by DBM	U^{DBM}_{N}	Utility of DBM under the strategy "Normal"
$Q_{\text{DepositLess}}$	Quantity of deposits below the expected amount to be included on-chain	U^{DBM}_{NPA}	Utility of DBM under the strategy "Not Packing All the deposits"
U^{DBM}_{PI}	Utility of DBM under the strategy "Packing Improperly"	R^{EBM}_{Include}	Revenue for EBM from including FHE public key and hashes of trained model parameters on-chain
R_{FHEM}	The revenue represented by the FHE-encrypted model decrypted by EBM	C^{EBM}_{RecFHEM}	Cost of process of receiving FHE-encrypted model
$C_{\text{GenFHEKey}}$	Cost of generating a pair of FHE keys including public key and secret key	Q_{HashM}	Quantity of hashes of model submitted by trainers
$\mathcal{R}_{\text{HashM}}$	Revenue for EBM per hash of model	V_{FHEM}	Version of FHE-encrypted model
V^{EBM}_{Now}	Version of the latest model owned by EBM just now before decrypting FHE-encrypted model	U^{EBM}_{N}	Utility of EBM under the strategy "Normal"
U^{EBM}_{NG}	Utility of EBM under the strategy "Not Generating" FHE keys	R^{TBM}_{Include}	Revenue for TBM from including testing data cases and hashes of FHE-encrypted models on-chain
$R_{\text{GenTDCases}}$	Revenue from generating testing data cases	$C_{\text{GenTDCases}}$	Cost of generating testing data cases
$Q_{\text{EncryptedM}}$	Quantity of hashes of FHE-encrypted models	$\mathcal{R}_{\text{EncryptedM}}$	Revenue for TBM per hash of Encrypted model
Q_{Cases}	Quantity of testing data cases submitted by TBM	$\mathcal{R}_{\text{Case}}$	Revenue for TBM per testing data case
$C^{\text{Unit}}_{\text{GenTDCase}}$	Cost of generating one unit of testing data case	U^{TBM}_{N}	Utility of TBM under the strategy "Normal"
U^{TBM}_{IT}	Utility of TBM under the strategy "Improper Testing data"	R^{SBM}_{Include}	Revenue for SBM from including Testing outputs $results$ and hashes of FHE-encrypted models on-chain
R_{Verify}	Revenue for SBM from verifying the performance of FHE-encrypted models using testing data cases	$C^{SBM}_{\text{RecFHEMs}}$	Cost of process of receiving FHE-encrypted models
C_{Verify}	Cost of verifying the performance of FHE-encrypted models	$Q_{\text{VerifiedM}}$	Quantity of verified FHE-encrypted models by SBM
$\mathcal{R}_{\text{VerifiedM}}$	Revenue for SBM per verified model from including performance of verified models on-chain	$\mathcal{R}_{\text{Verify}}$	Revenue for SBM per verified model and per testing data case from computing the performance of verified models
$C^{\text{Unit}}_{\text{Verify}}$	Cost per verified model and per case	U^{SBM}_{N}	Utility of SBM under the strategy "Normal"
U^{SBM}_{IRa}	Utility of SBM under the strategy "Improper Ranking"		

2) Problem Formulation and Incentive Mechanism Design: To maintain sustainability, the incentive of DeRelayL (e.g., mining reward, model weights) should at least satisfy **Individual Rationality (IR)** and **Incentive Compatibility (IC)** [4], [19], [49], [50]:

- *Individual Rationality:* All participants of the DeRelayL system should obtain a non-negative utility. Otherwise, the rational participants will not participate in the model training of the DeRelayL system.
- *Incentive Compatibility:* The incentive mechanism of the DeRelayL system should ensure that participants with normal behavior can obtain the maximum utility, which means that behaving normally is the optimal strategy for each participant.

Therefore, we will first formulate the utility (U) of each participant in the DeRelayL system based on the revenue (R) and cost (C). Key annotations are summarized in Table I.

(1) Model owner (MO): The utility of MO can be denoted:

$$\begin{aligned} U^{MO} &= R^{MO} - C^{MO} \\ &= R^{MO}_{\text{Now}} + R^{MO}_{\text{Future}} - C^{MO}_{\text{Deposit}} - C^{MO}_{\text{Transmit}} \end{aligned} \tag{1}$$

where R^{MO}_{Now} and R^{MO}_{Future} refer to the 4) additional citation reward as discussed in **Step (11)** of Section III-B. R^{MO}_{Now} is the citation reward of the current round, and R^{MO}_{Future} is the revenue of the future rounds, which will be calculated as a geometric series since the future revenue has a discount rate. For the cost, MO has deposit cost C^{MO}_{Deposit} for each round, but the cost is likely to be returned if the selected trainers behave normally. Moreover, MO also has transmission cost C^{MO}_{Transmit} when sending the model to the selected trainers.

(2) Trainer (T): The utility of T can be represented:

$$U^T = R^T - C^T = R^T_{\text{Now}} + R^T_{\text{Future}} \\ - C_{\text{Train}} - C^T_{\text{Deposit}} - C^T_{\text{RecM}} - C_{\text{Encrypt}} - C_{\text{Broadcast}} \tag{2}$$

where R^T_{Now} is the revenue of the current round, which contains the revenue of the received model from MO R^T_{RecM} and the revenue of the model trained by T R^T_{TrainedM}. And R^T_{Future} is the future revenue for additional citation reward, similar to MO. The cost of T consists of five parts: 1) model training cost C_{Train}; 2) deposit cost C^T_{Deposit}, which will be returned if behaving normally; 3) cost of receiving the model from MO C^T_{RecM}; 4) FHE encryption cost of trained model C_{Encrypt}; 5) cost of broadcasting encrypted model $C_{\text{Broadcast}}$.

(3) Deposit block miner (DBM): The utility of DBM is:

$$U^{DBM} = R^{DBM} - C^{DBM} \\ = R^{DBM}_{\text{Include}} - C_{\text{Mine}} \tag{3}$$

where R^{DBM}_{Include} is the incentive of miners to include deposit smart contracts as much as possible, so the revenue is proportional to the quantity of included data. C_{Mine} is the cost of mining the block, i.e., the computational cost of the PoW consensus model. Note that all miners (DBM, EBM, TBM, SBM) have the aforementioned R_{Include} and C_{Mine}. We also simply assume the block generation intervals are almost identical for all stages, thus the C_{Mine} is almost fixed.

(4) Encryption block miner (EBM): The utility of EBM can be formulated:

$$U^{EBM} = R^{EBM} - C^{EBM} \\ = R^{EBM}_{\text{Include}} + R_{\text{FHEM}} - C_{\text{Mine}} - C^{EBM}_{\text{RecFHEM}} - C_{\text{GenFHEKey}} \tag{4}$$

where R^{EBM}_{Include} is the incentive of including trained models' information, containing metadata and hash values. Since the EBM is responsible for generating the FHE key pair, the EBM can use the private key to decrypt encrypted models, as discussed in **Step (7)** of Section III-B. Thus, R_{FHEM} is the revenue for decrypting encrypted models, and C^{EBM}_{RecFHEM} is the cost for receiving the encrypted model (EB can only receive and decrypt the best one). C_{Mine} is the mining cost, and $C_{\text{GenFHEKey}}$ is the cost of generating the FHE key pair.

(5) Testing block miner (TBM): The utility of TBM is:

$$U^{TBM} = R^{TBM} - C^{TBM} \\ = R^{TBM}_{\text{Include}} + R_{\text{GenTDCases}} - C_{\text{Mine}} - C_{\text{GenTDCases}} \tag{5}$$

where R^{TBM}_{Include} is the incentive of including information of encrypted models using FHE, containing metadata and hash values. The TBMs are responsible for generating testing data, so they will be rewarded $R_{\text{GenTDCases}}$ according to the number of testing cases. Thus, there are corresponding costs of generating testing cases $C_{\text{GenTDCases}}$. Similar to other miners, C_{Mine} is the mining cost.

(6) Settlement block miner (SBM): The utility of SBM can be formulated:

$$U^{SBM} = R^{SBM} - C^{SBM} \\ = R^{SBM}_{\text{Include}} + R_{\text{Verify}} - C_{\text{Mine}} - C^{SBM}_{\text{RecFHEMs}} - C_{\text{Verify}} \tag{6}$$

where R^{SBM}_{Include} is the incentive of including verification confirmation details, containing metadata and performance index. The SBMs are responsible for verifying the performance of trained models, so they will be rewarded R_{Verify} according to the number of verified models, corresponding to the cost for receiving all encrypted models $C^{SBM}_{\text{RecFHEMs}}$ and verifying them C_{Verify}. C_{Mine} is the mining cost.

For participants, they have different strategies to choose from, which will lead to different utilities. We utilize "Normal (N)" to denote the participant behaves honestly following the procedure of the DeRelayL system. Specifically, MO may choose to not transmit the model to T (including transmitting fake weights), so the strategy set of MO is {Normal (N), Not Transmitting (NTm)}. For Ts, they may choose to not train the model (NTr) or not broadcast the trained model (NBr), so the strategy set of T is {Normal (N), Not Training (NTr), Not Broadcasting (NBr)}. The DBM may choose to pack partial deposit smart contracts (NPA) or pack improper ones (PI), thus the strategy set of DBM is {Normal (N), Not Packing All (NPA), Packing Improper Deposit Contracts (PI)}. Then, the EBM may not generate the FHE key (NG), so the strategy set of EBM is {Normal (N), Not Generating FHE Key (NG)}. For TBMs, they may upload improper testing cases (IT), thus the strategy set of TBM is {Normal (N), Improper Testing Cases (IT)}. Finally, the SBM may not rank the trained models properly (IRa), so the strategy set of SBM is {Normal (N), Improper Rank (IRa)}. The final utility expressions of each strategy are listed in Table II, and the detailed formulation, annotation, and explanation of each term can refer to Table I and Appendix A, available online. Note that, in this paper, we assume the knowledge used in model training is almost equal for every round (or satisfies the same distribution), so the model performance incremental from the current round/version to the next round/version is approximate. Therefore, we introduce Ⓒ to denote the value of the knowledge gap between two adjacent model versions, which is also the unit of measurement to unify the different values formulated in the DeRelayL system, as well as for issuing incentives (cryptocurrency/coin).

After formulating the utilities of each participant, we need to design an incentive mechanism to satisfy **IR** and **IC**. Specifically, to satisfy **IR**, the utilities of the "Normal" strategy should be no less than 0. According to the calculation in Appendix B, the reward ($\mathcal{R}$) of each block should satisfy the following condition set (**T1 - T8**):

T1: To guarantee **IR** of MO, we need to let $U^{MO}_N \geq 0$:

$$\mathcal{R}_{\text{Cited}} \geq \frac{(1-\beta)\cdot\left(Q_{\text{Selected}}\cdot(1-s)\cdot b^{MO} + k_{\text{Transmit}}\cdot|M|\right)}{\overline{Q^{MO}_{\text{Selected}}}} \tag{7}$$

TABLE II
UTILITIES OF DIFFERENT PARTICIPANTS' STRATEGIES

Participant	Strategy	Utility
MO	Normal (N)	$U_N^{MO} = \frac{\overline{Q^{MO}_{\text{Selected}}} \cdot \mathcal{R}_{\text{Cited}}}{1-\beta} - Q_{\text{Selected}} \cdot (1-s) \cdot b^{MO} - k_{\text{Transmit}} \cdot \lvert M \rvert$
	Not Transmitting (NTm)	$U_{NTm}^{MO} = -Q_{\text{Selected}} \cdot b^{MO}$
T	Normal (N)	$U_N^T = \left(V_{\text{RecM}} - V_{\text{Now}}^T + 1\right) \cdot \copyright + \frac{\overline{Q^{T}_{\text{Selected}}} \cdot \beta \cdot \mathcal{R}_{\text{Cited}}}{1-\beta}$ $- \left[P_{\text{Comp}} \cdot D \cdot \tau \cdot \lvert M \rvert + (1-s) \cdot b^T + k_{\text{Transmit}} \cdot \lvert M \rvert + k_{\text{Encrypt}} \cdot \lvert M \rvert + Q_{\text{Broadcast}} \cdot k_{\text{Transmit}} \cdot k_{\text{Expand}} \cdot \lvert M \rvert\right]$
	Not Training (NTr)	$U_{NTr}^T = (V_{\text{RecM}} - V_{\text{Now}}^T) \cdot \copyright - b^T - k_{\text{Transmit}} \cdot \lvert M \rvert$
	Not Broadcasting (NBr)	$U_{NBr}^T = \left(V_{\text{RecM}} - V_{\text{Now}}^T + 1\right) \cdot \copyright - \left[P_{\text{Comp}} \cdot D \cdot \tau \cdot \lvert M \rvert + b^T + k_{\text{Transmit}} \cdot \lvert M \rvert\right]$
DBM	Normal (N)	$U_N^{DBM} = Q_{\text{Deposit}} \cdot \mathcal{R}_{\text{Deposit}} - C_{\text{Mine}}$
	Not Packing All (NPA)	$U_{NPA}^{DBM} = Q_{\text{DepositLess}} \cdot \mathcal{R}_{\text{Deposit}} - C_{\text{Mine}}$
	Packing Improper Deposit Contracts (PI)	$U_{\text{PI}}^{DBM} = -C_{\text{Mine}}$
EBM	Normal (N)	$U_N^{EBM} = \left(Q_{\text{HashM}} \cdot \mathcal{R}_{\text{HashM}} + (V_{\text{FHEM}} - V_{\text{Now}}^{EBM}) \cdot \copyright\right) - \left(C_{\text{Mine}} + k_{\text{Transmit}} \cdot k_{\text{Expand}} \cdot \lvert M \rvert + C_{\text{GenFHEKey}}\right)$
	Not Generating FHE Key (NG)	$U_{NG}^{EBM} = -C_{\text{Mine}}$
TBM	Normal (N)	$U_N^{TBM} = Q_{\text{EncryptedM}} \cdot \mathcal{R}_{\text{EncryptedM}} + Q_{\text{Cases}} \cdot \mathcal{R}_{\text{Case}} - C_{\text{Mine}} - Q_{Cases} \cdot C_{\text{GenTDCase}}^{\text{Unit}}$
	Improper Testing Cases (IT)	$U_{IT}^{TBM} = -C_{\text{Mine}}$
SBM	Normal (N)	$U_N^{SBM} = Q_{\text{VerifiedM}} \cdot \mathcal{R}_{\text{VerifiedM}} + Q_{\text{VerifiedM}} \cdot Q_{\text{Cases}} \cdot \mathcal{R}_{\text{Verify}} - C_{\text{Mine}} - Q_{\text{VerifiedM}} \cdot k_{\text{Transmit}} \cdot k_{\text{Expand}} \cdot \lvert M \rvert$ $-Q_{\text{VerifiedM}} \cdot Q_{\text{Cases}} \cdot C_{\text{Verify}}^{\text{Unit}}$
	Improper Rank (IRa)	$U_{IRa}^{SBM} = -C_{\text{Mine}}$

T2: To guarantee **IR** of T, we need to let $U_N^T \geq 0$:

$$\mathcal{R}_{\text{Cited}} \geq \frac{(1-\beta)}{\overline{Q^T_{\text{Selected}}} \cdot \beta} \cdot \left(P_{\text{Comp}} \cdot D \cdot \tau \cdot |M| + (1-s) \cdot b^T + k_{\text{Transmit}} \cdot |M| + k_{\text{Encrypt}} \cdot |M| + Q_{\text{Broadcast}} \cdot k_{\text{Transmit}} \cdot k_{\text{Expand}} \cdot |M| - \left(V_{\text{RecM}} - V^T_{\text{Now}} + 1\right) \cdot \copyright\right) \tag{8}$$

T3: To guarantee **IR** of DBM, we need to let $U_N^{DBM} \geq 0$:

$$\mathcal{R}_{\text{Deposit}} \geq \frac{C_{\text{Mine}}}{Q_{\text{Deposit}}} \tag{9}$$

T4: To guarantee **IR** of EBM, we need to let $U_N^{EBM} \geq 0$:

$$\mathcal{R}_{\text{HashM}} \geq \frac{1}{Q_{\text{HashM}}} \cdot \left(C_{\text{Mine}} + k_{\text{Transmit}} \cdot k_{\text{Expand}} \cdot |M| + C_{\text{GenFHEKey}} - \left(V_{\text{FHEM}} - V^{EBM}_{\text{Now}}\right) \cdot \copyright\right) \tag{10}$$

T5: To guarantee **IR** of TBM, we need to let $U_N^{TBM} \geq 0$:

$$Q_{\text{EncryptedM}} \cdot \mathcal{R}_{\text{EncryptedM}} + Q_{\text{Cases}} \cdot \mathcal{R}_{\text{Case}} \geq C_{\text{Mine}} + Q_{Cases} \cdot C^{\text{Unit}}_{\text{GenTDCase}} \tag{11}$$

T6: To guarantee **IR** of SBM, we need to let $U_N^{SBM} \geq 0$:

$$Q_{\text{VerifiedM}} \cdot \mathcal{R}_{\text{VerifiedM}} + Q_{\text{VerifiedM}} \cdot Q_{\text{Cases}} \cdot \mathcal{R}_{\text{Verify}} \geq C_{\text{Mine}} + Q_{\text{VerifiedM}} \cdot k_{\text{Transmit}} \cdot k_{\text{Expand}} \cdot |M| + Q_{\text{VerifiedM}} \cdot Q_{\text{Cases}} \cdot C^{\text{Unit}}_{\text{Verify}} \tag{12}$$

To satisfy **IC**, the utilities of the "Normal" strategy should be greater than other strategies. According to Table II, some participants' other strategies have negative utilities, so they will choose "Normal" obviously. Specifically, trainer T requires additional constraints for the strategy of "Not Training" and "Not Broadcasting":

T7: For **IC** of T with "Not Training", there is a sufficient but not necessary condition that the deposit of T should not be lower than the value of the model (i.e., an effective deposit discussed in Section III-B). Otherwise, T will not have the motivation to train the model.

$$b^T > \left(V_{\text{RecM}} - V^T_{\text{Now}}\right) \cdot \copyright \tag{13}$$

T8: For **IC** of T with "Not Broadcasting", let $U_N^T - U_{NBr}^T > 0$, the condition can be formulated:

$$\mathcal{R}_{\text{Cited}} > \frac{1}{\overline{Q^T_{\text{Selected}}} \cdot \beta} \left((-s) \cdot b^T + k_{\text{Encrypt}} \cdot |M| + Q_{\text{Broadcast}} \cdot k_{\text{Transmit}} \cdot k_{\text{Expand}} \cdot |M|\right) \cdot (1-\beta) \tag{14}$$

Detailed derivation of **IR** and **IC** can refer to Appendix B, available online.

IV. NUMERICAL SIMULATIONS

To evaluate the effectiveness of the proposed DeRelayL system, we conduct a numerical simulation regarding bidding and matching of model owner and trainer, depositing, model training, performance ranking, block mining, as well as the corresponding incentive issuing. The numerical simulation mainly investigates two aspects, including sustainability and accessibility. Open-source simulation codes are available at https://github.com/Tengfei-Ma13206/DeRelayL_Simulation.

TABLE III
SIMULATION PARAMETERS SETTING

Simulation Parameter Description	Value
Total number of participants	256
Number of candidate miners in each round	128
Number of MOs and candidate Ts	128
Maximum number of Trainers cooperated by an MO	4
MO's budget	0.001 (coins)
Probability of successful model training by T	0.9
Number of testing data cases packaged by TBM	100
Proportion of Ts selected as successful model trainers	0.5
Reward base for miners, including $\mathcal{R}_{\text{Deposit}}$, $\mathcal{R}_{\text{HashM}}$, $\mathcal{R}_{\text{EncryptedM}}$, $\mathcal{R}_{\text{Case}}$, $\mathcal{R}_{\text{VerifiedM}}$, $\mathcal{R}_{\text{Verify}}$	0.001 (coins)

A. Experimental Settings and Procedure

In this subsection, we will discuss the experimental settings and procedure. To better explain the numerical simulation, we provide Table III to list the parameter settings. The detailed procedure of the numerical simulation is as follows:

(1) Role Allocation: The first step is to randomly allocate the roles of all participants, e.g., randomly selecting Ts. In this simulation, we assume that the total number of participants in the DeRelayL system is fixed at $Q_{\text{TotalParticipants}} = 256$. Among them, the number of miners in each round is fixed at $Q_{\text{Miners}} = 128$. The remaining participants are potential MOs and Ts, with the number being $Q_{\text{MO\&T}} = Q_{\text{TotalParticipants}} - Q_{\text{Miners}} = 128$. Specifically, the MOs were determined based on the previous round, since the MOs in this round were the Ts who luckily ranked at the top positions during the model performance evaluation in the last round. Moreover, for the cold start, we set that only the initiator of the genesis block serves as MO in the first round. Since each MO has limited capacity, we assume that they can only collaborate with up to $Q_{\text{SelectionLimit}} = 4$ Trainers in each round.

(2) Model Owner and Trainer Bidding: After determining the candidate numbers for each role, MOs and Ts start the bidding as discussed in Section III-C1. The MO's bidding strategy is based on a fixed budget, where we directly set $Budget_{\text{MO}} = 0.001$, representing the number of coins the MO possesses, denoted as $Q_{\text{CoinsOwnedByMO}}$. Therefore, the MO deposits for each Trainer can be calculated by $\frac{\min(Budget_{\text{MO}}, Q_{\text{CoinsOwnedByMO}})}{Q_{\text{SelectionLimit}}}$. Then, the bidding strategy for each T is based on the difference in model versions. We present the latest model version as V_{Latest} and T's current version as V^{T}_{Now}, and we assume the number of coins a T owns is $Q_{\text{CoinsOwnedByT}}$. Therefore, the bidding deposit is given by $\min(Q_{\text{CoinsOwnedByT}}, V_{\text{Latest}} - V^{T}_{\text{Now}} + 1)$, which means that the older the model version T has, the higher the motivation to place a larger bid. This setting accords the effective deposit (**T7** discussed in Section III-C2) to prevent Ts from receiving the latest model without training or broadcasting the model.

(3) Model Owner and Trainer Matching: After bidding, MO and T can be matched based on the deposit. We assume the matching process follows a simple greedy algorithm discussed in Section III-C1. In this case, MO will sort Ts in descending order based on the model performance from the previous round, and T will sort MOs based on the number of deposit coins. The first MO selects the top $Q_{\text{SelectionLimit}}$ Ts, the second MO selects the next $Q_{\text{SelectionLimit}}$ Ts, and so on, until all MOs or Ts have found a match participant.

(4) Success of Model Training: After MO and T matching, a miner is randomly selected from the previously grouped miners to complete the PoW mining and is designated as DBM (Note that the "random selection" of the miner is to simulate the winner of PoW mining, rather than random selection in practical implementation). The DBM includes the deposit records into the DeRelayL blockchain. Afterward, Ts involved in the deposit start to train the model. To simulate the potential accidents that might happen in practice, we assume that Ts have a probability of $Pr_{\text{Training}} = 0.9$ to successfully train a new model, while the remaining Ts may fail to train the model due to various reasons.

(5) Performance Ranking: An EBM is randomly selected from the miners to include FHE keys to the DeRelayL blockchain. Following this, a randomly selected TBM includes a fixed number ($Q_{\text{Cases}} = 100$) of testing data cases into the DeRelayL blockchain. Similarly, a randomly selected SBM includes the testing results to the DeRelayL blockchain. Finally, from those who successfully trained new models, we assume that a fixed proportion of Ts ($s = 0.5$), with flooring applied, are selected to be regarded as successful trainers (i.e., rank in the top positions). Also note that the "random selection" of the miners is to simulate the winner of PoW mining, rather than random selection in practical implementation.

(6) Incentive Issuing: Based on step **(5)**, the DeRelayL system will issue rewards to the MO and all predecessor MOs up to the genesis block owner (additional citation reward as discussed in **Step (11)** of Section III-B), each receiving one ©, where every successful model selection triggers this reward. Moreover, the corresponding incentives (such as mining reward, testing data reward, FHE key generation reward, etc.) will also be settled in this step. Note that, in this simulation, the base reward for miners is set at 0.001 coins, and we simply assume that no one adopts strategies other than "Normal" due to the IR and IC ensured in Section III-C2, because the mistakes would lead to forking by honest participants, which will be more effective to exclude them from the simulation in this study. In future research, the consequences of forking in DeRelayL is a promising topic that can be further investigated to discuss its unique attributes and effects.

B. Sustainability

The core motivation of the proposed DeRelayL is to build a sustainable decentralized learning system, which means that the design of a sustainable training system must first ensure that it remains operational, i.e., participants have incentives to continue their involvement. In detail, a participant's willingness to be involved largely depends on their estimated future rewards. Under stable conditions, with no abrupt changes in rules or participant behavior, their future rewards can be estimated using historical data. Therefore, we measure past rewards in terms of coins accumulated, as the cost of each training round is roughly constant. Then, we plot the changes in coin quantity for each participant as rounds progressed in Fig. 3. The observed trend

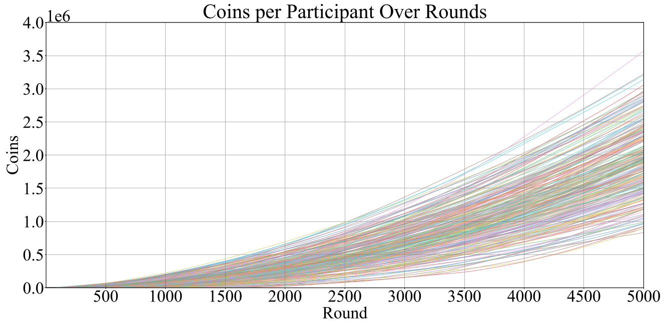

Fig. 3. Coins per participants over rounds.

indicates that the growth rate of coins accelerates over time, demonstrating the sustainability of the proposed DeRelayL, where all participants have positive estimated future rewards to continue their involvement. To further explain this result, we provide a simple proof:

As discussed in Section IV-A, we assume that model versions are iteratively updated by selecting from all participants in a round-robin fashion, and every participant successfully uploads their trained model periodically. Simply assume that there are $Q_{\text{Participants}}$ participants, denoted as $P_1, P_2, \ldots, P_{i_p}, \ldots, P_{Q_{\text{Participants}}}$. Moreover, we consider a single DeRelayL blockchain where models are represented by $M_1, M_2, \ldots, M_{i_m}, \ldots, M_{Q_{\text{Model}}}$, where each successive model is trained based on the previous one.

We focus only on the additional citation reward as discussed in **Step (11)** of Section III-B, which has the greatest impact on coin accumulation. Due to the IR and IC ensured in Section III-C2, we assume all participants train diligently. Thus, we map each participant P_{i_p} to models $M_{i_p}, M_{i_p+Q_{\text{Participants}}}, M_{i_p+2\cdot Q_{\text{Participants}}}, \ldots$, and the mapping relationship satisfies $i_m \ mod \ Q_{\text{Participants}} = i_p$.

When participant P_{i_p} trains and uploads a model for the x-th time, the number of coins Q_{Coins} they receive is:

$$\begin{aligned} Q_{\text{Coins}} &= Q_{\text{Participants}} + 2\cdot Q_{\text{Participants}} + \cdots + (x-1)\cdot Q_{\text{Participants}} \\ &= \frac{x(x-1)Q_{\text{Participants}}}{2} = O(x^2) = O((i_m)^2) \end{aligned} \tag{15}$$

where the round number here equals $i_m = i_p + (x-1)\cdot Q_{\text{Participants}}$. Therefore, if every participant actively engages in training, their Q_{Coins} grows quadratically with the number of rounds, implying that the coin accumulation rate accelerates over time. Detailed annotations can refer to Table IV.

C. Accessibility

Besides the sustainability of DeRelayL, it is also crucial to ensure participants can obtain the trained models, denoted by accessibility, which fits the motivation that participants can train and share the model together. To analyze the accessibility, we plot the model version distribution over rounds, as shown in Fig. 4. In this figure, we illustrate the percentage of participants who possess the models of the latest 10 versions (“Latest-v” in Fig. 4 denotes the model of the v-th version before the current version), older versions, or none. Note that the performance of models within the same version is approximately consistent from a global view. Fig. 4 illustrates that all participants can possess models after rounds of training, and their owned models are kept updated following the system operation. Moreover, after convergence, the possession percentage of model versions tends to be stable, because the trainers possessing older models will tend to bid higher to compete for the training opportunity, while those possessing recent models may have less incentive to compete against them. Therefore, the models will naturally be distributed as evenly as possible among all participants. To explore this phenomenon, we also conduct a brief proof:

TABLE IV
KEY ANNOTATIONS FOR THE NUMERICAL SIMULATION
(IN THE ORDER OF APPEARANCE)

Variable	Description
$Q_{\text{TotalParticipants}}$	Total number of participants
Q_{Miners}	Number of candidate miners in each round
$Q_{\text{MO\&T}}$	Number of MOs and candidate Ts
$Q_{\text{SelectionLimit}}$	Maximum number of Trainers cooperated by an MO
$Budget_{\text{MO}}$	MO’s budget
$Q_{\text{CoinsOwnedByMO}}$	Number of coins owned by MO
V_{Latest}	Latest model version
V^T_{Now}	Current model version owned by T
$Q_{\text{CoinsOwnedByT}}$	Number of coins owned by T
Pr_{Training}	Probability of successful model training by T
Q_{Cases}	Number of testing data cases packaged by TBM
s	Proportion of Ts selected as successful model trainers
$Q_{\text{Participants}}$	Total number of participants
P_{i_p}	Participant indexed by i_p.
M_{i_m}	Model version of the i_m-th training round
Q_{Model}	Number of models
i_m	Model index of training round
i_p	Participant index of training round
x	The x-th time the participant P_{i_p} trains and uploads a model
Q_{Coins}	Number of coins received by a participant
Q_{LastMO}	Number of MOs in the previous (last) round
Q_{LastT}	Number of Trainers in the previous (last) round
Q_{MO}	Number of MOs in the current round
Q_{T}	Number of Trainers in the current round
N_r	Number of Trainers in round r

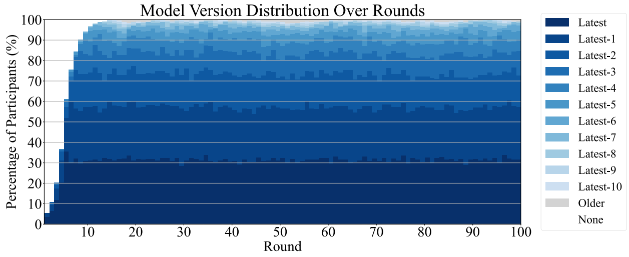

Fig. 4. Model version distribution over rounds.

We denote the number of MOs from the previous round as Q_{LastMO}, and the corresponding number of trainers is:

$$Q_{\text{LastT}} = \min\left(Q_{MO\&T} - Q_{\text{LastMO}}, Q_{\text{LastMO}} \cdot Q_{\text{SelectionLimit}}\right) \tag{16}$$

In the current round, the number of MOs is:

$$\begin{aligned} Q_{\text{MO}} &= s \cdot Q_{\text{LastT}} \\ &= s \cdot \left(\min\left(Q_{MO\&T} - Q_{\text{LastMO}}, Q_{\text{LastMO}} \cdot Q_{\text{SelectionLimit}}\right)\right) \end{aligned} \tag{17}$$

And the number of trainers in the current round is:

$$Q_{\text{T}} = \min\left(Q_{MO\&T} - Q_{\text{MO}}, Q_{\text{MO}} \cdot Q_{\text{SelectionLimit}}\right) \tag{18}$$

Since $Q_{MO\&T}$ is fixed, and Q_{T} initially grows but is eventually bounded by $Q_{MO\&T}$. Therefore, after convergence, we can obtain:

$$Q_{\text{T}} = Q_{MO\&T} - Q_{\text{MO}} = Q_{MO\&T} - s \cdot Q_{\text{LastT}} \tag{19}$$

Let the number of trainers in round r be N_r, we can obtain:

$$\begin{aligned} N_{r+1} &= Q_{MO\&T} - s \cdot N_r \\ &= Q_{MO\&T} \cdot \left(1 + (-s) + (-s)^2 + \cdots + (-s)^{i-1}\right) \\ &\quad + (-s)^i \cdot N_r \end{aligned} \tag{20}$$

Since $0 < s < 1$ is the proportion of Ts selected as successful model trainers, as i tends to infinity, we can obtain:

$$N_{r+i} = Q_{MO\&T} \cdot \frac{1}{1+s} = \frac{Q_{MO\&T}}{1+s} \tag{21}$$

where $\frac{Q_{MO\&T}}{1+s}$ is also fixed, indicating that the number of trainers remains stable at this value across the training rounds, thus the phenomenon observed in Fig. 4 can be demonstrated. Furthermore, the fluctuations in Fig. 4 are due to the probabilistic settings of each trainer to simulate unexpected failures of training. Detailed annotations can refer to Table IV.

V. Discussion

By further clarifying the details of the proposed DeRelayL paradigm, we selected some representative and concerning points to conduct a comprehensive discussion in this section.

A. System Anonymity

In most public blockchain systems (e.g., Bitcoin [47]), anonymity is a key feature designed to mitigate the risk of a 51% attack, which means an entity controlling over 50% of the network nodes could potentially alter the blockchain, compromising its tamper-proof nature. However, in DeRelayL, the necessity for strict anonymity seems to be less critical due to the distinct target of the DeRelayL blockchain compared to general public blockchains. Specifically, the primary value of public blockchains lies in cryptocurrency, and any breach of consensus (such as a 51% attack) would lead to a collapse in value. In contrast, the DeRelayL blockchain's value resides in the trained models, meaning that a 51% attack would not yield additional profit. Even if an attacker gains control over the DeRelayL blockchain, they cannot meaningfully promote the model improvement, since the actual training is conducted by independent third-party trainers. In this case, the implications of a 51% attack differ from those in public blockchains. For example, if such an attack occurs during $Round_n$, the model performance published in $Round_{n-1}$ remains unaffected, because it is an objective measure that is independent of blockchain consensus. For subsequent rounds, recognizing the 51% attack, honest participants can fork the blockchain or create a new blockchain-based on the model from $Round_{n-1}$, ensuring the continuity of the latest models without any loss (detailed discussion on forking is provided in Section V-B).

From another perspective, the DeRelayL system is adaptable to different blockchain architectures. In this paper, we utilize a PoW consensus model to enable permissionless participation in model training and system maintenance. However, DeRelayL can also be deployed on a consortium blockchain, where participants are pre-approved by a governing committee. This verification process may eliminate the anonymity of participants, supervising honest behaviors. In this case, some steps in the DeRelayL workflow can be simplified. For instance, a consortium blockchain could employ consensus mechanisms other than PoW to reduce computational costs associated with block mining. Additionally, approved participants could transmit plaintext model data directly, which will also effectively reduce the resource overhead discussed in Section VI-A.

In summary, the DeRelayL paradigm does not strictly emphasize anonymity during system operation and can be adapted to various scenarios through appropriate modifications.

B. Blockchain Forking

As mentioned in Section V-A, the DeRelayL blockchain is designed to support flexible forking in the event of attacks or unexpected issues. Therefore, it is important to consider the potential impact that the forking of the DeRelayL blockchain will result in. In general public blockchains (e.g., Bitcoin [47]), forking leads to the creation of subchains, and miners must decide which subchain to follow. In theory, miners will tend to follow the longest valid subchain (or longest valid chain), which represents the most computational workers (in PoW systems) or the greatest stake (in Proof of Stake (PoS) systems). This rule helps protect the blockchain against attacks such as double spending and ensures that honest participants who follow the protocol can prevail over malicious users.

In the case of DeRelayL, normal operations will not be affected by forking in terms of model performance improvement. This is because the common model training process is inherently iterative, and model performance naturally fluctuates across training epochs. Generally, the training rounds in DeRelayL can be viewed similarly, where each forking represents an exploration of different potential paths toward a global optimum, and each block within a subchain represents a local optimum along that journey. Additionally, when considering the longest valid subchain, the risk of double spending seems to have minimal impact on the normal operation of the DeRelayL system. For instance, if the DeRelayL system allows forking in $Round_n$, malicious users might attempt to double spend by depositing in one subchain with one model owner and in another subchain with a different model owner, aiming to exchange the same coins for multiple models. It seems that malicious users can obtain many models through the double spending attack. However, the models they obtain will be from the same version of $Round_n$, which will have comparable performance from a global aspect. Furthermore, the dishonest behavior will be recorded on the blockchain, and their deposits will not be returned, preventing the attackers from joining in subsequent training rounds and obtaining newer models.

Thus, the forking of DeRelayL exhibits different attributes compared to general public blockchains. However, to seriously investigate the consequences, it is necessary to conduct a dedicated study focusing on its unique features and effects.

C. Necessity of the Four-Stage Process

In this paper, we propose a four-stage process of the DeRelayL, including Deposit Block (DB), Encryption Block (EB), Testing Block (TB), and Settlement Block (SB), which is designed to ensure the integrity, consistency, fairness, and security of the training and evaluation process. We consider that each stage serves a distinct function that cannot be combined without compromising the system's objectives. The detailed schemes will be discussed as follows:

(1) Combination of DB and EB: In the DB stage, both the trainers and model owners are committed to the process, preventing dishonest behaviors like the trainers stealing the model weights or the model owners failing to provide the model weights. If the DB miner and EB miner were the same entity, the model would be registered after passing through the DB process, but this could result in inconsistencies between the tested and the trained models. In this case, the separation of DB and EB ensures that the model's hash value is determined independently, preventing any tampering or unintended changes. On the other hand, the EB stage is initiated after DB to enable the selection of a new miner if needed, ensuring that the process remains decentralized and fair. Thus, if DB and EB were combined, it would be harder to replace a dishonest participant, leading to potential manipulation.

(2) Combination of EB and TB: EB and TB cannot be combined because the testing data cannot be released during the EB stage before the model's hash ($Hash(M_n^k)$) is fixed. Allowing this would risk overfitting, as the model would have access to the test data before the final evaluation, which means that the model trainers can use the testing data to fine-tune their models so that they can have better evaluation scores. Moreover, the TB stage also prevents unfair testing data cases by separating the training and testing phases, ensuring that the testing data is unbiased, which is crucial for fair evaluation.

(3) Combination of TB and SB: If TB and SB were combined, it would be possible for a malicious participant to manipulate the testing data to artificially boost a specific model's ranking. To mitigate this, the PoW mechanism [47] randomly selects the TB miner, ensuring that no single entity can control the process in the long term. Moreover, the SB stage serves to prevent unfair performance testing, ensuring that the model's evaluation is independent and accurate. Additionally, waiting until the model training is completed before the TB and SB phases helps avoid cheating, such as manipulating testing cases before evaluation. Therefore, the fact the same TB miner and SB miner could allow one entity to potentially manipulate both the test and score phases.

In summary, the four-stage process is essential for maintaining fairness, preventing dishonest behavior, and ensuring the integrity of the system.

D. Consensus Mechanism

As mentioned in Section III-B, we utilize the Proof of Work (PoW) [47] consensus model for block generation as an example by default. In our system, PoW is primarily used to maintain the randomness of miner selection, which helps prevent collusion and ensures the integrity of the consensus process. However, other consensus mechanisms capable of maintaining similar randomness can also be employed, while PoW is preferred due to its well-understood and established properties. The computational difficulty of PoW in this context can be adjusted by modifying the difficulty of solving the cryptographic puzzle [51]. In this case, the adjustment allows the system to control the time taken for block generation, ensuring consistency in the block creation process. It is worth noting that, a key distinction from Bitcoin's PoW is that, while Bitcoin's PoW is primarily used to control the time interval between blocks, the focus in DeRelayL is more on maintaining randomness in the selection of participants for training. This setup reduces the likelihood of malicious actors benefiting from any potential system collapse, ensuring that no participant gains at the expense of others.

On the other hand, the primary purpose of PoW in this system is to randomly select a miner, but ensuring the security of the system against attacks involves more than just this random selection, such as the 51% attack [52]. A 51% attack would typically occur if a malicious entity controls the majority of the network's computational power and can alter the consensus [52]. However, in DeRelayL, as long as the majority of miners are honest, even if a dishonest group successfully mines a block, their blocks will be discarded by the network. This is because the system relies on the assumption that the majority of participants will act honestly and that any block mined by dishonest individuals will eventually be rejected by the honest majority. Furthermore, the integrity of the system depends on the assumption that the honest majority will always outweigh any dishonest minority. If a dishonest majority were to emerge, they would likely be phased out over time as the system evolves, especially given that participants are incentivized to act honestly in order to receive rewards or maintain their reputation within the network. The decentralized and distributed nature of the blockchain, along with the transparent consensus mechanism, ensures that any attempt to subvert the system by dishonest actors is self-limiting, as their blocks would not be accepted by the majority of honest miners. In fact, to further strengthen the security issues, additional mechanisms, such as staking or reputation systems [53], could be introduced to discourage dishonest behavior and incentivize honest participation, ensuring that the network remains secure and trustworthy even in the event of potential attacks.

E. Potential Real-World Applications

Currently, large model training mainly relies on crawling corpora from the Internet (e.g., models like GPT-3 [54] and Llama 3.1 [55]), while the data/knowledge contributors cannot share the profits of large models. This approach leads to a significant drawback: the model trainers find it extremely difficult to obtain data that is not available online, while the data contributors are not willing to contribute knowledge to the Internet. A real-world case has occurred that some artists have refused to share their artworks with model trainers [4], further highlighting the limitations of relying on Internet-based data collection. Therefore, DeRelayL's potential impact lies in creating a decentralized training paradigm. By allowing the exchange of model usage rights for data, DeRelayL intends to boost the richness of training data. In this case, a wider variety of data from different offline sources can be integrated into the training process, such as specialized industry datasets, personal diaries, and private research findings. Since this mechanism does not require a large amount of capital injection to purchase data, it reduces the economic cost of further enhancing the performance of large models. Moreover, the cost-effective approach also makes it more accessible for a broader range of participants, including small-scale research teams and individual developers, to contribute to and benefit from the development of large models. Therefore, the DeRelayL system can not only increase individual influence on the development of large models but also help prevent large companies from monopolizing the values embedded in large models.

VI. Limitations and Future Research Topics

However, it is necessary to point out that the DeRelayL is still in its early stage, and there are several limitations that remain to be completely addressed in the future.

A. Resource Overhead

In this paper, we employ fully homomorphic encryption (FHE) [45] to transmit model weights, where FHE is an encryption scheme that enables analytical functions to be run directly on encrypted data while yielding the same encrypted results as if the functions were run on plaintext data [46]. The purpose of utilizing FHE is to evaluate the performance of trained models without exposing the model weights. However, our theoretical framework assumes an ideal scenario where the computational, memory, and storage overhead of FHE is acceptable. In practice, high computation and memory overhead make FHE computation over $10,000\times$ slower than unencrypted computation on conventional computing systems that process unencrypted data [46]. This substantial overhead also increases the cost of transmitting encrypted models, as it requires significantly more memory, storage, and internet traffic. Additionally, FHE computations may introduce small errors that accumulate over FHE operations performed, leading to approximate rather than precise results [45]. Consequently, FHE is less suitable for applications requiring high numerical precision, such as scientific computations, due to its reliance on polynomial approximations.

Therefore, to implement the DeRelayL system effectively, several challenges must be addressed: (1) The computational, memory, and storage overhead of FHE needs to be reduced, potentially through the development of more effective FHE algorithms. While specialized hardware can significantly accelerate FHE operations, it may not be accessible to common users, limiting their ability to participate in the DeRelayL system. (2) Due to the storage demands of FHE, broadcasting encrypted models poses a challenge for common users. A potential solution is that trainers can upload encrypted models to decentralized storage, and then broadcast only the storage addresses to other participants, avoiding the significant transmission costs associated with P2P communication. (3) The trade-off between FHE resource overhead and the precision required for performance evaluation should be further explored, since it is intuitive that reducing numerical precision could simplify FHE operations, thereby decreasing resource consumption. (4) Alternative methodologies for evaluating the training process should also be investigated. The motivation for using FHE is to prevent model weight leakage during performance evaluation. Thus, we consider that the appropriate utilization of technologies like zero-knowledge proof (ZKP) may also provide solutions to the proposed scenario of DeRelayL. ZKP is a cryptographic method that allows one party (i.e., the trainer) to prove to another party (i.e., the SBM) that they know a piece of information (i.e., the performance of a trained model) without revealing the actual information itself (e.g., the exact model weight) [56], [57]. Additionally, other cryptographic techniques that achieve similar objectives can also be considered to improve the DeRelayL system.

B. Model Size Dilemma

In this paper, we assume that the model size (or specific model architecture) is fixed at the creation of the genesis block. This assumption is reasonable, since a fixed model can standardize the programming interface for each participant, lowering the barrier to joining the DeRelayL system. However, according to scaling laws [2], [3], model size influences the upper-performance limit. Therefore, over many rounds of training, a fixed model size will eventually reach its performance ceiling, where further training yields diminishing returns. In this case, the DeRelayL system meets the model size dilemma, where model improvements are minimal no matter how to conduct the continued training, making the resource costs associated with training unworthy. Consequently, our mathematical modeling of DeRelayL does not consider a stop condition when the cost of model training exceeds the incremental value gained from the model improvement.

Additionally, it is necessary to establish mechanisms for dynamically enlarging the model size without initiating a new DeRelayL blockchain. Several critical challenges need to be addressed: determining who will be responsible for managing model size enlargement, identifying the appropriate timing/block/round for implementing the larger model, and defining the optimal size for the new model. Furthermore, the performance change associated with model enlargement must be carefully evaluated, and suitable technologies must be identified for transferring knowledge from the previous model to the new one. Correspondingly, the responsibility for executing this knowledge transfer and evaluating its performance must also be clearly defined. Therefore, the dynamic adjustment of model size remains an open research topic in the DeRelayL.

In practice, the model size dilemma is an extreme case that would only append after a significant number of training rounds, which would require a considerable amount of time to achieve. Theoretically, over such a long period, other

DeRelayL blockchains with higher expected performance would emerge, attracting rational participants to migrate to these new blockchains. However, this transition would lead to another dilemma: a senior participant in the old DeRelayL blockchain would become a freshman in the new one, which makes the accumulated contribution of the senior participant in the old blockchain useless. Therefore, the model size dilemma of the DeRelayL paradigm and its associated challenges are worthy of further research and investigation.

C. Training Time and Training Data Dilemma

In the DeRelayL system, the training time for each round is dynamically adjusted to optimize training efficiency and performance. The duration is automatically set based on the minimum time required to achieve a positive increment in model performance. If the training time is too long, it may indicate that the training process is inefficient, leading to unnecessary resource consumption. On the contrary, if the duration is too short, there is a risk that the model's performance may not improve, resulting in the efforts of the trainers ineffective. To this end, the system should prioritize the final outcome, the real growth in performance, rather than focusing too heavily on the specifics of the training process itself. We consider that the "valuable" data can be evaluated by whether it could quickly improve the model's performance. For example, for large datasets, the system encourages trainers to break their datasets into smaller, more manageable parts, each of which can contribute to rapid performance gains. This approach may ensure that trainers with larger datasets can participate multiple times and continue to benefit from the model's improvements. Therefore, how to adjust the training duration according to performance increments and data value is an important challenge, which influences the fairness and efficiency of collaborative training in the DeRelayL system.

Besides the training itself, we also notice that the true value of utilized data may not be immediately reflected in the model's performance in the current round, which may discourage participants who possess high-quality data from contributing. Therefore, how to effectively reflect the contribution of the participants can be further studied in the future, e.g., by applying data valuation-related approaches [58].

VII. Conclusion

In this paper, we propose a novel collaborative learning paradigm, named Decentralized Relay Learning (DeRelayL), a sustainable decentralized learning system where permissionless participants can contribute to model training in a relay-like manner and share the model together. We introduce the architecture and workflow of DeRelayL and incentive mechanisms to ensure sustainability. Moreover, theoretical analysis and numerical simulations are conducted to demonstrate the effectiveness of the proposed DeRelayL. At last, we provide comprehensive discussions of DeRelayL regarding promising research topics in the future. In summary, the proposed DeRelayL training mechanism aims to solve the challenge of motivating widespread participation in large-scale model training, especially by encouraging individuals to contribute data that is not readily available on the Internet. If this mechanism operates effectively, it could lead to a more equitable distribution of the benefits derived from AI, where participants actively influence and benefit from the intelligent Big Data era they help create. We expect that our insights can inspire related studies into decentralized collaborative learning systems that empower common users, fostering a fairer, more sustainable digital ecosystem, where data creators have greater control and can benefit from the AI models they help develop.

References

[1] Y. Chang et al., "A survey on evaluation of large language models," *ACM Trans. Intell. Syst. Technol.*, vol. 15, no. 3, pp. 1–45, 2024.

[2] J. Kaplan et al., "Scaling laws for neural language models," 2020, *arXiv:2001.08361*.

[3] B. Zhang, Z. Liu, C. Cherry, and O. Firat, "When scaling meets LLM finetuning: The effect of data, model and finetuning method," in *Proc. 12th Int. Conf. Learn. Representations*, 2024, pp. 1–20.

[4] H. Duan, A. El Saddik, and W. Cai, "Incentive mechanism design toward a win–win situation for generative art trainers and artists," *IEEE Trans. Computat. Social Syst.*, vol. 11, no. 6, pp. 7528–7540, Dec. 2024.

[5] A. Alami, R. Pardo, and J. Linåker, "Free open source communities sustainability: Does it make a difference in software quality?," *Empir. Softw. Eng.*, vol. 29, no. 5, 2024, Art. no. 114.

[6] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Int. Conf. Artif. Intell. Statist.*, PMLR, 2017, pp. 1273–1282.

[7] M. Ruan, Y. Li, W. Zhang, L. Song, and W. Xu, "Optimal power control for over-the-air federated learning with gradient compression," in *Proc. IEEE 30th Int. Conf. Parallel Distrib. Syst. (ICPADS)*, IEEE, 2024, pp. 326–333.

[8] K. Hsieh, A. Phanishayee, O. Mutlu, and P. Gibbons, "The non-IID data quagmire of decentralized machine learning," in *Proc. Int. Conf. Mach. Learn.*, PMLR, 2020, pp. 4387–4398.

[9] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh, "SCAFFOLD: Stochastic controlled averaging for federated learning," in *Proc. Int. Conf. Mach. Learn.*, PMLR, 2020, pp. 5132–5143.

[10] E. T. M. Beltrán et al., "Decentralized federated learning: Fundamentals, state of the art, frameworks, trends, and challenges," *IEEE Commun. Surveys Tuts.*, vol. 25, no. 4, pp. 2983–3013, Fourth Quarter 2023.

[11] Y. Qu, M. P. Uddin, C. Gan, Y. Xiang, L. Gao, and J. Yearwood, "Blockchain-enabled federated learning: A survey," *ACM Comput. Surv.*, vol. 55, no. 4, pp. 1–35, 2022.

[12] L. He, A. Bian, and M. Jaggi, "COLA: Decentralized linear learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 4541–4551.

[13] S. Li, T. Zhou, X. Tian, and D. Tao, "Learning to collaborate in decentralized learning of personalized models," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 9766–9775.

[14] H. Kim, J. Park, M. Bennis, and S.-L. Kim, "Blockchained on-device federated learning," *IEEE Commun. Lett.*, vol. 24, no. 6, pp. 1279–1283, Jun. 2020.

[15] Y. Li, C. Chen, N. Liu, H. Huang, Z. Zheng, and Q. Yan, "A blockchain-based decentralized federated learning framework with committee consensus," *IEEE Netw.*, vol. 35, no. 1, pp. 234–241, Jan./Feb. 2021.

[16] X. Qu, S. Wang, Q. Hu, and X. Cheng, "Proof of federated learning: A novel energy-recycling consensus algorithm," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 8, pp. 2074–2085, Aug. 2021.

[17] P. Ramanan and K. Nakayama, "BAFFLE : Blockchain Based Aggregator Free Federated Learning," in *Proc. IEEE Int. Conf. Blockchain (Blockchain)*, IEEE, 2020, pp. 72–81.

[18] Y. J. Kim and C. S. Hong, "Blockchain-based node-aware dynamic weighting methods for improving federated learning performance," in *Proc. 20th Asia-Pacific Netw. Oper. Manage. Symp. (APNOMS)*, IEEE, 2019, pp. 1–4.

[19] S. Fan, H. Zhang, Y. Zeng, and W. Cai, "Hybrid blockchain-based resource trading system for federated learning in edge computing," *IEEE Internet Things J.*, vol. 8, no. 4, pp. 2252–2264, Feb. 2021.

[20] J. Li et al., "Blockchain assisted decentralized federated learning (BLADE-FL): Performance analysis and resource allocation," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 10, pp. 2401–2415, Oct. 2022.

[21] Z. Qin, X. Yan, M. Zhou, and S. Deng, “BlockDFL: A blockchain-based fully decentralized peer-to-peer federated learning framework,” in *Proc. ACM Web Conf.*, 2024, pp. 2914–2925.

[22] A. Ekuban and J. Domingue, “Towards decentralised learning analytics (positioning paper),” in *Proc. ACM Web Conf.*, 2023, pp. 1435–1438.

[23] J. Zhang, Y. Wu, and R. Pan, “Incentive mechanism for horizontal federated learning based on reputation and reverse auction,” in *Proc. Web Conf.*, 2021, pp. 947–956.

[24] M. Diskin et al., “Distributed deep learning in open collaborations,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, pp. 7879–7897.

[25] M. Ryabinin and A. Gusev, “Towards crowdsourced training of large neural networks using decentralized mixture-of-experts,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 3659–3672.

[26] W. Zheng, R. Deng, W. Chen, R. A. Popa, A. Panda, and I. Stoica, “Cerebro: A platform for multi-party cryptographic collaborative learning,” in *Proc. 30th USENIX Secur. Symp.*, 2021, pp. 2723–2740.

[27] J. Kang et al., “Tiny multi-agent DRL for twins migration in UAV metaverses: A multi-leader multi-follower stackelberg game approach,” *IEEE Internet Things J.*, vol. 11, no. 12, pp. 21021–21036, Jun. 2024.

[28] Y. Gao, Z. Song, and J. Yin, “GradientCoin: A peer-to-peer decentralized large language models,” The University of Washington, 2023, *arXiv:2308.10502*.

[29] Z.-H. Bo et al., “Relay learning: A physically secure framework for clinical multi-site deep learning,” *NPJ Digit. Med.*, vol. 6, no. 1, 2023, Art. no. 204.

[30] Y. Fraboni, R. Vidal, L. Kameni, and M. Lorenzi, “Clustered sampling: Low-variance and improved representativity for clients selection in federated learning,” in *Proc. Int. Conf. Mach. Learn.*, PMLR, 2021, pp. 3407–3416.

[31] H. Wang, M. Yurochkin, Y. Sun, D. Papailiopoulos, and Y. Khazaeni, “Federated learning with matched averaging,” in *Proc. Int. Conf. Learn. Representations*, 2020, pp. 1–16.

[32] T. Lin, L. Kong, S. U. Stich, and M. Jaggi, “Ensemble distillation for robust model fusion in federated learning,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 2351–2363.

[33] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, “Federated optimization in heterogeneous networks,” *Proc. Mach. Learn. Syst.*, vol. 2, pp. 429–450, 2020.

[34] H.-Y. Chen and W.-L. Chao, “FedBE: Making Bayesian model ensemble applicable to federated learning,” in *Proc. Int. Conf. Learn. Representations*, 2021, pp. 1–21.

[35] Q. Chen, Z. Wang, J. Hu, H. Yan, J. Zhou, and X. Lin, “PAGE: Equilibrate personalization and generalization in federated learning,” in *Proc. ACM Web Conf.*, 2024, pp. 2955–2964.

[36] M. Zhou, G. Liu, K. Lu, R. Mao, and H. Liao, “Accelerating the decentralized federated learning via manipulating edges,” in *Proc. ACM Web Conf.*, 2024, pp. 2945–2954.

[37] K. Wang, Q. He, F. Chen, H. Jin, and Y. Yang, “FedEdge: Accelerating edge-assisted federated learning,” in *Proc. ACM Web Conf.*, 2023, pp. 2895–2904.

[38] Y. Shi, H. Duan, L. Yang, and W. Cai, “An energy-efficient and privacy-aware decomposition framework for edge-assisted federated learning,” *ACM Trans. Sensor Netw.*, vol. 18, no. 4, pp. 1–24, 2022.

[39] Y. Zhang et al., “Privacy-preserving and fairness-aware federated learning for critical infrastructure protection and resilience,” in *Proc. ACM Web Conf.*, 2024, pp. 2986–2997.

[40] M. Cao, L. Zhang, and B. Cao, “Toward on-device federated learning: A direct acyclic graph-based blockchain approach,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 4, pp. 2028–2042, Apr. 2023.

[41] B. Buyukates et al., “Proof-of-contribution-based design for collaborative machine learning on blockchain,” in *Proc. IEEE Int. Conf. Decentralized Appl. Infrastructures (DAPPS)*, IEEE, 2023, pp. 13–22.

[42] E. Ebrahimi, M. Sober, A.-T. Hoang, C. U. Ileri, W. Sanders, and S. Schulte, “Blockchain-based federated learning utilizing zero-knowledge proofs for verifiable training and aggregation,” in *Proc. IEEE Int. Conf. Blockchain (Blockchain)*, IEEE, 2024, pp. 54–63.

[43] H. Yazdaninejad, M. Rajarajan, and M. Krol, “A blockchain-enabled and transparent evaluation of ML models in the decentralised marketplace,” in *Proc. IEEE Int. Conf. Blockchain (Blockchain)*, IEEE, 2024, pp. 458–463.

[44] Q. Yang, Y. Liu, T. Chen, and Y. Tong, “Federated machine learning: Concept and applications,” *ACM Trans. Intell. Syst. Technol. (TIST)*, vol. 10, no. 2, pp. 1–19, 2019.

[45] P. Martins, L. Sousa, and A. Mariano, “A survey on fully homomorphic encryption: An engineering perspective,” *ACM Comput. Surv. (CSUR)*, vol. 50, no. 6, pp. 1–33, 2017.

[46] J. Kim, S. Kim, J. Choi, J. Park, D. Kim, and J. H. Ahn, “SHARP: A short-word hierarchical accelerator for robust and practical fully homomorphic encryption,” in *Proc. 50th Annu. Int. Symp. Comput. Archit.*, 2023, pp. 1–15.

[47] S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system,” Satoshi Nakamoto, 2008.

[48] D. Friedman, “The double auction market institution: A survey,” in *The Double Auction Market*, Evanston, IL, USA: Routledge, 2018, pp. 3–26.

[49] S. Paris, F. Martignon, I. Filippini, and L. Chen, “An efficient auction-based mechanism for mobile data offloading,” *IEEE Trans. Mobile Comput.*, vol. 14, no. 8, pp. 1573–1586, Aug. 2015.

[50] P. Sun, X. Chen, G. Liao, and J. Huang, “A profit-maximizing model marketplace with differentially private federated learning,” in *Proc. IEEE Conf. Comput. Commun.*, IEEE, 2022, pp. 1439–1448.

[51] W. Cai, Z. Wang, J. B. Ernst, Z. Hong, C. Feng, and V. C. Leung, “Decentralized applications: The blockchain-empowered software system,” *IEEE Access*, vol. 6, pp. 53019–53033, 2018.

[52] C. Ye, G. Li, H. Cai, Y. Gu, and A. Fukuda, “Analysis of security in blockchain: Case study in 51% -attack detecting,” in *Proc. 5th Int. Conf. Dependable Syst. Appl. (DSA)*, IEEE, 2018, pp. 15–24.

[53] Z. Zhou, M. Wang, C.-N. Yang, Z. Fu, X. Sun, and Q. J. Wu, “Blockchain-based decentralized reputation system in E-commerce environment,” *Future Gener. Comput. Syst.*, vol. 124, pp. 155–167, 2021.

[54] T. Brown et al., “Language models are few-shot learners,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 1877–1901.

[55] A. Dubey et al., “The llama 3 herd of models,” 2024, *arXiv:2407.21783*.

[56] T. Liu, X. Xie, and Y. Zhang, “ZkCNN: Zero knowledge proofs for convolutional neural network predictions and accuracy,” in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2021, pp. 2968–2985.

[57] C. Weng, K. Yang, X. Xie, J. Katz, and X. Wang, “Mystique: Efficient conversions for zero-knowledge proofs with applications to machine learning,” in *Proc. 30th USENIX Secur. Symp.*, 2021, pp. 501–518.

[58] X. Shi and H. Duan, “Data valuation and pricing in Internet of Things: Survey and vision,” in *Proc. IEEE Int. Conf. Smart Internet Things (SmartIoT)*, IEEE, 2024, pp. 547–554.

Haihan Duan (Member, IEEE) received his BEng degree in computer science and technology from East China Normal University, Shanghai, China, in 2017, and the MEng degree in software engineering from Sichuan University, Chengdu, China, in 2020, and the PhD degree in computer and information engineering from The Chinese University of Hong Kong, Shenzhen, China, in 2023. He is currently an Associate professor with Artificial Intelligence Research Institute, Shenzhen MSU-BIT University (SMBU), and he is also with Guangdong-Hong Kong-Macao Joint Laboratory for Emotion Intelligence and Pervasive Computing, Shenzhen, China. Before joining SMBU, he worked as a postdoctoral research fellow with the University of Ottawa and Mohamed bin Zayed University of Artificial Intelligence (MBZUAI), located in Abu Dhabi, UAE. His research interests include multimedia, blockchain and Web3, metaverse, human-centered computing, and medical image analysis.

Tengfei Ma is currently working toward the BEng (forth-year) degree in computer engineering with the School of Science and Engineering, The Chinese University of Hong Kong, Shenzhen, China. He is also a visiting student with the Artificial Intelligence Research Institute, Shenzhen MSU-BIT University, and with the Guangdong- Hong Kong-Macao Joint Laboratory for Emotion Intelligence and Pervasive Computing, Shenzhen, China. His research interests include blockchain, retrieval augmented generation, and game theory.

Yuyang Qin is currently working toward the BSc (third-year) degree in data science with the School of Data Science, The Chinese University of Hong Kong, Shenzhen, China. He is also a visiting student with the Artificial Intelligence Research Institute, Shenzhen MSU-BIT University, and with the Guangdong-Hong Kong-Macao Joint Laboratory for Emotion Intelligence and Pervasive Computing, Shenzhen, China. His research interests include blockchain development, data analysis, and blockchain system design.

Runhao Zeng (Member, IEEE) received the PhD degree in software engineering from the South China University of Technology, in 2021. He is currently an Associate professor with Artificial Intelligence Research Institute, Shenzhen MSU-BIT University (SMBU), and he is also with Guangdong-Hong Kong-Macao Joint Laboratory for Emotion Intelligence and Pervasive Computing, Shenzhen, China. He has authored or coauthored several peer-reviewed papers on computer vision, machine learning on top-tier conferences and journals, including the Proceedings of NeurIPS, CVPR, ICCV, and TPAMI. His research interests include machine learning, computer vision, with particular focus on video analysis.

Wei Cai (Senior Member, IEEE) received the BEng in software engineering from Xiamen University in 2008, the MSc in electrical engineering and computer science from Seoul National University in 2011, and the PhD degree in electrical and computer engineering from The University of British Columbia in 2016. He is currently a tenure-track Assistant professor of computer science and systems with the School of Engineering and Technology, University of Washington, Tacoma, WA, USA. He is now leading the Decentralized Computing Laboratory., Prior to joining UW, he was an Assistant professor of electrical and computer engineering with The Chinese University of Hong Kong, Shenzhen, China. He has also conducted research visits with Academia Sinica Taiwan, The Hong Kong Polytechnic University, and the National Institute of Informatics Japan. He has coauthored more than 100 peer-reviewed journal and conference papers and has received 6 Best Paper Awards. His research interest include decentralized computing, with emphasis on mechanism design, social computing, multimedia, and applications. He was an Associate editor for *ACM Transactions on Multimedia Computing, Communications, and Applications* and *IEEE Transactions on Computational Social Systems*, and previously for *IEEE Transactions on Cloud Computing*. Dr. Cai is a Steering Committee member for ACM NOSSDAV, where he was a TPC co-chair in 2023, and has been an Area Chair for ACM MM since 2023. He is a member of ACM.

Victor C. M. Leung (Life Fellow, IEEE) is currently with Artificial Intelligence Research Institute, Shenzhen MSU-BIT University, Shenzhen, China. He is also an emeritus professor of electrical and computer engineering and the Director of the Laboratory for Wireless Networks and Mobile Systems, The University of British Columbia (UBC), Canada. His research interests include wireless networks and mobile systems. He has published widely in these areas. He was the recipient of 1977 APEBC Gold Medal, 1977–1981 NSERC Postgraduate Scholarships, IEEE Vancouver Section Centennial Award, 2011 UBC Killam Research Prize, 2017 Canadian Award for Telecommunications Research, 2018 IEEE TCGCC Distinguished Technical Achievement Recognition Award, and 2018 ACM MSWiM Reginald Fessenden Award. He has coauthored papers that won the 2017 IEEE ComSoc Fred W. Ellersick Prize, the 2017 IEEE Systems Journal Best Paper Award, the 2018 IEEE CSIM Best Journal Paper Award, and the 2019 IEEE TCGCC Best Journal Paper Award. He has been serving on the editorial boards of *IEEE Transactions on Green Communications and Networking*, *IEEE Transactions on Cloud Computing*, *IEEE Access*, *IEEE Network*, and several other journals. He is named in the current Clarivate Analytics list of “Highly Cited Researchers.” He is a fellow of the Royal Society of Canada (Academy of Science), Canadian Academy of Engineering, and Engineering Institute of Canada.

Xiping Hu (Member, IEEE) received the PhD degree from the University of British Columbia, Vancouver, BC, Canada. He is currently a professor with the Beijing Institute of Technology, and with Shenzhen MSU-BIT University, China. He has authored or coauthored more than 150 papers published and presented in prestigious conferences and journals, such as IEEE TPAMI/TMC/TPDS/TIP/JSAC, IEEE COMST, ACM MobiCom/MM/SIGIR/WWW, AAAI, and IJCAI. He has been an associate editor of IEEE TCSS, and the lead guest editors of IEEE IoT Journal and IEEE TASE etc. He has been granted several key national research projects as Principal investigator. He was the Co-Founder and CTO of Bravolol Ltd., Hong Kong, a leading language learning mobile application company with over 100 million users, and listed as the top 2 language education platform globally. His research areas consist of mobile cyber-physical systems, crowd sensing and affective computing.

Appendix

A. Utility Formulation

This section presents the detailed derivation regarding the utilities of different participants' strategies, and the final formulations are listed in Table II. Since the formulation involves lots of parameters, we summarize the key annotations listed in Table I for better understanding. The utility formulation is following with Section III-C2, and the extended derivation is shown as follows:

(1) Model owner (MO): The utility of MO can be denoted as:

$$\begin{aligned} U^{MO} &= R^{MO} - C^{MO} \\ &= R^{MO}_{\text{Now}} + R^{MO}_{\text{Future}} - C^{MO}_{\text{Deposit}} - C^{MO}_{\text{Transmit}} \end{aligned} \tag{22}$$

where R^{MO}_{Now} and R^{MO}_{Future} refer to the 4) additional citation reward as discussed in **Step (11)** of Section III-B. R^{MO}_{Now} is the citation reward of the current round, and R^{MO}_{Future} is the revenue of the future rounds, which will be calculated as a geometric series since the future revenue has a discount rate. For the cost, MO has deposit cost C^{MO}_{Deposit} for each round, but the cost is likely to be returned if the selected trainers behave normally. Moreover, MO also has transmission cost C^{MO}_{Transmit} when sending the model to the selected trainers.

The strategy set of MO is {Normal (N), Not Transmitting (NTm)}, where "Not Transmitting (NTm)" is an abstract presentation of dishonest behavior, including transmitting fake weights, etc. The utilities of different strategies are formulated as follows:

(1.1) MO with N:

The revenue of MO with N is given by:

$$\begin{aligned} R^{MO} &= R^{MO}_{\text{Now}} + R^{MO}_{\text{Future}} \\ &= \overline{Q^{MO}_{\text{Selected}}} \cdot \mathcal{R}_{\text{Cited}} + \frac{\overline{Q^{MO}_{\text{Selected}}} \cdot \beta \cdot \mathcal{R}_{\text{Cited}}}{1-\beta} \\ &= \frac{\overline{Q^{MO}_{\text{Selected}}} \cdot \mathcal{R}_{\text{Cited}}}{1-\beta} \end{aligned} \tag{23}$$

where $\overline{Q^{MO}_{\text{Selected}}}$ is the number of selected models of MO, and $\mathcal{R}_{\text{Cited}}$ is the revenue for citation reward. Kindly remind that we designed a mechanism to reward the citation of good models, discussed in **4) additional citation reward** of **Step (11)** in Section III-B, and it should have a discount rate $0 \leq \beta \leq 1$ when calculating the future revenue. In summary, the final formula of R^{MO} is presented as the sum of geometric series, i.e., the total expected revenue for citation reward in now and future.

The cost of MO with N is given by:

$$\begin{aligned} C^{MO} &= C^{MO}_{\text{Deposit}} + C^{MO}_{\text{Transmit}} \\ &= Q_{\text{Selected}} \cdot (1-s) \cdot b^{MO} + k_{\text{Transmit}} \cdot |M| \end{aligned} \tag{24}$$

where Q_{Selected} is number of selected trainers and b^{MO} is MO's deposit cost for one trainer, determined by Algorithm 1 in Section III-C1. And $0 \leq s \leq 1$ is the proportion of selected qualified models. Kindly remind that, to alleviate the lazy workers, we design a mechanism that only the trained models which have performance ranking in top-$\mathcal{K}$ can return the deposit cost for both original model owners in **3) returning qualified deposit** of **Step (11)** in Section III-B. Here, we use $0 \leq s \leq 1$ to denote the proportion that can pass the performance comparison for better generalization. Therefore, $Q_{\text{Selected}} \cdot (1-s) \cdot b^{MO}$ is the expected deposit cost of MO. Moreover, k_{Transmit} is the transmission cost coefficient, and $|M|$ is the number of parameters in the model, so $k_{\text{Transmit}} \cdot |M|$ can denote the transmission cost.

Therefore, the utility of MO with N can be formulated as:

$$\begin{aligned} U^{MO}_{N} &= R^{MO} - C^{MO} \\ &= \frac{\overline{Q^{MO}_{\text{Selected}}} \cdot \mathcal{R}_{\text{Cited}}}{1-\beta} - Q_{\text{Selected}} \cdot (1-s) \cdot b^{MO} \\ &\quad - k_{\text{Transmit}} \cdot |M| \end{aligned} \tag{25}$$

(1.2) MO with NTm:

The revenue of MO with NTm is given by:

$$R^{MO} = 0 \tag{26}$$

which means T cannot obtain original models to train, so MO also cannot obtain any citation reward.

The cost of MO with NTm is given by:

$$C^{MO} = C^{MO}_{\text{DepositLoss}} = Q_{\text{Selected}} \cdot b^{MO} \tag{27}$$

which means that the deposit of MO is b^{MO} will lose, if the MO behave dishonestly. Moreover, since the MO have selected Q_{Selected} Ts, the deposit loss will be $Q_{\text{Selected}} \cdot b^{MO}$.

Therefore, the utility of MO with NTm can be formulated as:

$$U^{MO}_{NTm} = -Q_{\text{Selected}} \cdot b^{MO} \tag{28}$$

(2) Trainer (T): The utility of T can be represented as:

$$\begin{aligned} U^{T} &= R^{T} - C^{T} = R^{T}_{\text{Now}} + R^{T}_{\text{Future}} \\ &\quad - C_{\text{Train}} - C^{T}_{\text{Deposit}} - C^{T}_{\text{RecM}} - C_{\text{Encrypt}} - C_{\text{Broadcast}} \end{aligned} \tag{29}$$

where R^{T}_{Now} is the revenue of the current round, which contains the revenue of the received model from MO R^{T}_{RecM} and the revenue of the model trained by T R^{T}_{TrainedM}. And R^{T}_{Future} is the future revenue for additional citation reward, similar to MO. The cost of T consists of five parts: 1) model training cost C_{Train}; 2) deposit cost C^{T}_{Deposit}, which will be returned if behaving normally; 3) cost of receiving the model from MO C^{T}_{RecM}; 4) FHE encryption cost of trained model C_{Encrypt}; 5) cost of broadcasting encrypted model $C_{\text{Broadcast}}$.

The T may choose to not train the model (NTr) or not broadcast the trained model (NBr), so the strategy set of T is {Normal (N), Not Training (NTr), Not Broadcasting (NBr)}. The utilities of different strategies are formulated as follows:

(2.1) T with N:

The revenue of T with N is given by:

$$\begin{aligned} R^{T} &= R^{T}_{\text{Now}} + R^{T}_{\text{Future}} \\ &= (R_{\text{RecM}} + R_{\text{TrainedM}}) + R^{T}_{\text{Cited}} \\ &= [(V_{\text{RecM}} - V^{T}_{\text{Now}}) \cdot \textcircled{C} + 1 \cdot \textcircled{C}] + \frac{\overline{Q^{T}_{\text{Selected}}} \cdot \beta \cdot \mathcal{R}_{Cited}}{1-\beta} \\ &= \big(V_{\text{RecM}} - V^{T}_{\text{Now}} + 1\big) \cdot \textcircled{C} + \frac{\overline{Q^{T}_{\text{Selected}}} \cdot \beta \cdot \mathcal{R}_{Cited}}{1-\beta} \end{aligned} \tag{30}$$

where $V_{\text{RecM}} - V_{\text{Now}}^{T}$ denotes the version gap between the received model (the latest model) and the model that T has already owned (the out-of-date model), and we introduce $\textcircled{C}$ to denote the value of knowledge gap between two adjacent model versions. The second term is the additional citation reward, similar to the MO, which is presented as the sum of geometric series based on the discount rate $0 \leq \beta \leq 1$.

The cost of T with N is given by:

$$\begin{aligned} C^T &= C_{\text{Train}} + C_{\text{Deposit}}^{T} + C_{\text{RecM}}^{T} + C_{\text{Encrypt}} + C_{\text{Broadcast}} \\ &= P_{\text{Comp}} \cdot D \cdot \tau \cdot |M| + (1-s) \cdot b^T \\ &\quad + k_{\text{Transmit}} \cdot |M| + k_{\text{Encrypt}} \cdot |M| \\ &\quad + Q_{\text{Broadcast}} \cdot k_{\text{Transmit}} \cdot k_{\text{Expand}} \cdot |M| \end{aligned} \tag{31}$$

where P_{Comp} is the price of computational resource, D represents the training data size, τ is the training duration, and $|M|$ denotes the number of parameters in the model, so the first term is the training cost. Then, $0 \leq s \leq 1$ denotes the proportion that can pass the performance comparison, thus $(1-s) \cdot b^T$ is the expected deposit cost. Moreover, k_{Transmit} is the transmission cost coefficient, k_{Encrypt} is the FHE encryption cost coefficient, so the transmission cost and encryption cost can be presented. At last, $Q_{\text{Broadcast}}$ is the number of transmissions in broadcasting, k_{Expand} is the expanding coefficient of model parameter number after FHE, and the aforementioned parameters can calculate the broadcasting cost.

Therefore, the utility of T with N can be formulated as:

$$\begin{aligned} U_N^T &= R^T - C^T \\ &= \big(V_{\text{RecM}} - V_{\text{Now}}^{T} + 1\big) \cdot \textcircled{C} + \frac{\overline{Q_{\text{Selected}}^{T}} \cdot \beta \cdot \mathcal{R}_{\text{Cited}}}{1-\beta} \\ &\quad - \big[P_{\text{Comp}} \cdot D \cdot \tau \cdot |M| + (1-s) \cdot b^T + k_{\text{Transmit}} \cdot |M| \\ &\quad + k_{\text{Encrypt}} \cdot |M| + Q_{\text{Broadcast}} \cdot k_{\text{Transmit}} \cdot k_{\text{Expand}} \cdot |M|\big] \end{aligned} \tag{32}$$

(2.2) T with NTr:

The revenue of T with NTr is given by:

$$\begin{aligned} R^T &= R_{\text{RecM}} \\ &= (V_{\text{RecM}} - V_{\text{Now}}^{T}) \cdot \textcircled{C} \end{aligned} \tag{33}$$

where $(V_{\text{RecM}} - V_{\text{Now}}^{T}) \cdot \textcircled{C}$ is the value of the version gap between the received model (the latest model) and the model that T has already owned (the out-of-date model).

The cost of T with NTr is given by:

$$\begin{aligned} C^T &= C_{\text{DepositLost}}^{T} + C_{\text{RecM}}^{T} \\ &= b^T + k_{\text{Transmit}} \cdot |M| \end{aligned} \tag{34}$$

where b^T is the deposit of T, which will be lost since the T has failed to train. Moreover, the T has the transmission cost $k_{\text{Transmit}} \cdot |M|$ for receiving the model from MO.

Therefore, the utility of T with NTr can be formulated as:

$$\begin{aligned} U_{NTr}^T &= R^T - C^T \\ &= (V_{\text{RecM}} - V_{\text{Now}}^{T}) \cdot \textcircled{C} - b^T - k_{\text{Transmit}} \cdot |M| \end{aligned} \tag{35}$$

(2.3) T with NBr:

The revenue of T with NBr is given by:

$$\begin{aligned} R^T &= R_{\text{RecM}} + R_{\text{TrainedM}} \\ &= (V_{\text{RecM}} - V_{\text{Now}}^{T}) \cdot \textcircled{C} + 1 \cdot \textcircled{C} \\ &= \big(V_{\text{RecM}} - V_{\text{Now}}^{T} + 1\big) \cdot \textcircled{C} \end{aligned} \tag{36}$$

where the Ts have finished the model training, so they will have one more $\textcircled{C}$ compared with "Not Training".

The cost of T with NBr is given by:

$$\begin{aligned} C^T &= C_{\text{Train}} + C_{\text{DepositLost}}^{T} + C_{\text{RecM}}^{T} \\ &= P_{\text{Comp}} \cdot D \cdot \tau \cdot |M| + b^T + k_{\text{Transmit}} \cdot |M| \end{aligned} \tag{37}$$

which has the training cost and transmission cost for receiving the model from MO, and the deposit of T b^T will also be lost.

Therefore, the utility of T with NBr can be formulated as:

$$\begin{aligned} U_{NBr}^T &= R^T - C^T \\ &= \big(V_{\text{RecM}} - V_{\text{Now}}^{T} + 1\big) \cdot \textcircled{C} \\ &\quad - \big[P_{\text{Comp}} \cdot D \cdot \tau \cdot |M| + b^T + k_{\text{Transmit}} \cdot |M|\big] \end{aligned} \tag{38}$$

(3) Deposit block miner (DBM): The utility of DBM is:

$$\begin{aligned} U^{DBM} &= R^{DBM} - C^{DBM} \\ &= R_{\text{Include}}^{DBM} - C_{\text{Mine}} \end{aligned} \tag{39}$$

where R_{Include}^{DBM} is the incentive of miners to include deposit smart contracts as much as possible, so the revenue is proportional to the quantity of included data. C_{Mine} is the cost of mining the block, i.e., the computational cost of the PoW consensus model. Note that, all miners (DBM, EBM, TBM, SBM) have the aforementioned R_{Include} and C_{Mine}. We also simply assume the block generation intervals are almost identical for all stages, thus the C_{Mine} is almost fixed.

The DBM may choose to pack partial deposit smart contracts (NPA) or pack improper ones (PI), thus the strategy set of DBM is $\{$Normal (N), Not Packing All (NPA), Packing Improper Deposit Contracts (PI)$\}$. The utilities of different strategies are formulated as follows:

(3.1) DBM with N:

The revenue of DBM with N is given by:

$$\begin{aligned} R^{DBM} &= R_{\text{Include}}^{DBM} \\ &= Q_{\text{Deposit}} \cdot \mathcal{R}_{\text{Deposit}} \end{aligned} \tag{40}$$

where Q_{Deposit} is the number of deposit smart contracts, and $\mathcal{R}_{\text{Deposit}}$ is the revenue to incentivize the DBM to pack deposit smart contracts as much as possible. Under this setting, a problem has emerged: what if a dishonest DBM packs invalid smart contracts to cheat for additional revenue? Due to the decentralized consensus of blockchain, other miners will check the validity of packed content, so the dishonest DBM will be recognized as "Packing Improper Deposit Contracts". In this case, other miners will build a new fork to invalidate the block from a dishonest DBM, thus the corresponding dishonest DBM will lose all revenues.

The cost of DBM with N is given by:

$$C^{DBM} = C_{\text{Mine}} \tag{41}$$

where C_{Mine} is the mining cost. Remind that all miners (DBM, EBM, TBM, SBM) have the C_{Mine}, and we simply assume the block generation intervals are almost identical for all stages, thus the C_{Mine} is almost fixed to all miners.

Therefore, the utility of DBM with N can be formulated as:

$$\begin{aligned} U_N^{DBM} &= R^{DBM} - C^{DBM} \\ &= Q_{\text{Deposit}} \cdot \mathcal{R}_{\text{Deposit}} - C_{\text{Mine}} \end{aligned} \tag{42}$$

(3.2) DBM with NPA:

The revenue of DBM with NPA is given by:

$$\begin{aligned} R^{DBM} &= R_{\text{Include}}^{DBM} \\ &= Q_{\text{DepositLess}} \cdot \mathcal{R}_{\text{Deposit}} \end{aligned} \tag{43}$$

where $0 < Q_{\text{DepositLess}} < Q_{\text{Deposit}}$, which means the DBM has not packed all deposit smart contracts that broadcast in the DeRelayL network, and $Q_{\text{DepositLess}} \cdot \mathcal{R}_{\text{Deposit}}$ is the including revenue.

The cost of DBM with NPA is given by:

$$C^{DBM} = C_{\text{Mine}} \tag{44}$$

where C_{Mine} is the mining cost.

Therefore, the utility of DBM with NPA can be formulated as:

$$\begin{aligned} U_{NPA}^{DBM} &= R^{DBM} - C^{DBM} \\ &= Q_{\text{DepositLess}} \cdot \mathcal{R}_{\text{Deposit}} - C_{\text{Mine}} \end{aligned} \tag{45}$$

(3.3) DBM with PI:

The revenue of DBM with PI is given by:

$$R^{DBM} = 0 \tag{46}$$

where the dishonest DBM will not obtain revenue since the cheating behavior will be recognized by other miners in the DeRelayL system.

The cost of DBM with PI is given by:

$$C^{DBM} = C_{\text{Mine}} \tag{47}$$

where C_{Mine} is the mining cost.

Therefore, the utility of DBM with PI is given by:

$$U_{\text{PI}}^{DBM} = -C_{\text{Mine}} \tag{48}$$

(4) Encryption block miner (EBM): The utility of EBM is:

$$\begin{aligned} U^{EBM} &= R^{EBM} - C^{EBM} \\ &= R_{\text{Include}}^{EBM} + R_{\text{FHEM}} - C_{\text{Mine}} - C_{\text{RecFHEM}}^{EBM} - C_{\text{GenFHEKey}} \end{aligned} \tag{49}$$

where R_{Include}^{EBM} is the incentive of including trained models' information, containing metadata and hash values. Since the EBM is responsible for generating the FHE key pair, the EBM can use the private key to decrypt encrypted models, as discussed in **Step (7)** of Section III-B. Thus, R_{FHEM} is the revenue for decrypting encrypted models of FHE, and C_{RecFHEM}^{EBM} is the cost for receiving the encrypted model (EB can just receive the model with best performance). C_{Mine} is the mining cost, and $C_{\text{GenFHEKey}}$ is the cost of generating FHE key pair.

The EBM may not generate an FHE key (NG) or send a random number to disturb the training system. Thus, the strategy set of EBM is $\{$Normal (N), Not Generating FHE Key (NG)$\}$. The utilities of different strategies are formulated as follows:

(4.1) EBM with N:

The revenue of EBM with N is given by:

$$\begin{aligned} R^{EBM} &= R_{\text{Include}}^{EBM} + R_{\text{FHEM}} \\ &= Q_{\text{HashM}} \cdot \mathcal{R}_{\text{HashM}} + (V_{\text{FHEM}} - V_{\text{Now}}^{EBM}) \cdot \textcircled{C} \end{aligned} \tag{50}$$

where Q_{HashM} is the number of packed hash values of trained models, and $\mathcal{R}_{\text{HashM}}$ is the corresponding revenue. The second term is a special revenue for the EBM, as discussed in **Step (7)** of Section III-B, where $(V_{\text{FHEM}} - V_{\text{Now}}^{EBM}) \cdot \textcircled{C}$ is the value of the version gap between the encrypted model (there are lots of encrypted models are broadcast to the network, but the EBM will tend to decrypt the model that ranks at top-1 during the performance evaluation) and the model that the EBM has already owned (out-of-date model).

The cost of EBM with N is given by:

$$\begin{aligned} C^{EBM} &= C_{\text{Mine}} + C_{\text{RecFHEM}}^{EBM} + C_{\text{GenFHEKey}} \\ &= C_{\text{Mine}} + k_{\text{Transmit}} \cdot k_{\text{Expand}} \cdot |M| + C_{\text{GenFHEKey}} \end{aligned} \tag{51}$$

where C_{Mine} is the mining cost. If the EBMs want to obtain the latest model, they will afford the transmission cost for receiving the encrypted model, denoting as $k_{\text{Transmit}} \cdot k_{\text{Expand}} \cdot |M|$, where k_{Transmit} is the transmission cost coefficient, k_{Expand} is the expanding coefficient of model parameter number after FHE, and $|M|$ denotes the number of parameters in the model.

Therefore, the utility of EBM with N can be formulated as:

$$\begin{aligned} U_N^{EBM} &= R^{EBM} - C^{EBM} \\ &= \big(Q_{\text{HashM}} \cdot \mathcal{R}_{\text{HashM}} + (V_{\text{FHEM}} - V_{\text{Now}}^{EBM}) \cdot \textcircled{C}\big) \\ &\quad - (C_{\text{Mine}} + k_{\text{Transmit}} \cdot k_{\text{Expand}} \cdot |M| + C_{\text{GenFHEKey}}) \end{aligned} \tag{52}$$

(4.2) EBM with NG:

The revenue of EBM with NG is given by:

$$R^{EBM} = 0 \tag{53}$$

where the revenue of dishonest EBM will be equal to zero. This is because the invalid FHE key pair (e.g., randomly generated numbers) will be recognized by other participants, and rational participants will build a new fork to invalidate the block from a dishonest EBM.

The cost of EBM with NG is given by:

$$C^{EBM} = C_{\text{Mine}} \tag{54}$$

where C_{Mine} is the mining cost.

Therefore, the utility of EBM with NG can be formulated as:

$$U_{NG}^{EBM} = -C_{\text{Mine}} \tag{55}$$

(5) Testing block miner (TBM): The utility of TBM is:

$$\begin{aligned} U^{TBM} &= R^{TBM} - C^{TBM} \\ &= R_{\text{Include}}^{TBM} + R_{\text{GenTDCases}} - C_{\text{Mine}} - C_{\text{GenTDCases}} \end{aligned} \tag{56}$$

where R_{Include}^{TBM} is the incentive of including information of encrypted models using FHE, containing metadata and hash values. The TBMs are responsible for generating testing data, so they will be rewarded $R_{\text{GenTDCases}}$ according to the number of testing cases. Thus, there are corresponding costs of generating testing cases $C_{\text{GenTDCases}}$. Similar to other miners, C_{Mine} is the mining cost.

For TBMs, they may upload improper testing cases (IT), thus the strategy set of TBM is {Normal (N), Improper Testing Cases (IT)}. The utilities of different strategies are formulated as follows:

(5.1) TBM with N:

The revenue of TBM with N is given by:

$$\begin{aligned} R^{TBM} &= R^{TBM}_{\text{Include}} + R_{\text{GenTDCases}} \\ &= Q_{\text{EncryptedM}} \cdot \mathcal{R}_{\text{EncryptedM}} + Q_{\text{Cases}} \cdot \mathcal{R}_{\text{Case}} \end{aligned} \tag{57}$$

where $Q_{\text{EncryptedM}}$ is the number of packed hash values of encrypted trained models after FHE, and $\mathcal{R}_{\text{EncryptedM}}$ is the corresponding incentive for including the hash values. Moreover, Q_{Cases} is the number of included testing data cases, and $\mathcal{R}_{\text{Case}}$ is the corresponding incentive for generating the testing data cases.

The cost of TBM with N is given by:

$$\begin{aligned} C^{TBM} &= C_{\text{Mine}} + C_{\text{GenTDCases}} \\ &= C_{\text{Mine}} + Q_{Cases} \cdot C^{\text{Unit}}_{\text{GenTDCase}} \end{aligned} \tag{58}$$

where C_{Mine} is the mining cost. Moreover, the generation, preparation, or collection of testing data also has cost, and we use $C^{\text{Unit}}_{\text{GenTDCase}}$ to denote the generation cost of the testing data per unit/case, thus $Q_{Cases} \cdot C^{\text{Unit}}_{\text{GenTDCase}}$ is the total cost for generating the testing data cases.

Therefore, the utility of TBM with N can be formulated as:

$$\begin{aligned} U^{TBM}_{N} &= R^{TBM} - C^{TBM} \\ &= Q_{\text{EncryptedM}} \cdot \mathcal{R}_{\text{EncryptedM}} + Q_{\text{Cases}} \cdot \mathcal{R}_{\text{Case}} \\ &\quad - C_{\text{Mine}} - Q_{Cases} \cdot C^{\text{Unit}}_{\text{GenTDCase}} \end{aligned} \tag{59}$$

(5.2) TBM with IT:

The revenue of TBM with IT is given by:

$$R^{TBM} = 0 \tag{60}$$

where the revenue of dishonest TBM will be equal to zero. This is because a dishonest TBM will be recognized by most participants since the testing data is accessible to the public that every participant can check, thus the rational participants will build a new fork to invalidate the block from a dishonest TBM.

The cost of TBM with IT is given by:

$$C^{TBM} = C_{\text{Mine}} \tag{61}$$

where the dishonest TBMs only have the mining cost C_{Mine}, since they will not truly prepare the testing data or just utilize old testing data uploaded by other TBMs in the previous training rounds.

Therefore, the utility of TBM with IT can be formulated as:

$$U^{TBM}_{IT} = -C_{\text{Mine}} \tag{62}$$

(6) Settlement block miner (SBM): The utility of SBM is:

$$\begin{aligned} U^{SBM} &= R^{SBM} - C^{SBM} \\ &= R^{SBM}_{\text{Include}} + R_{\text{Verify}} - C_{\text{Mine}} - C^{SBM}_{\text{RecFHEMs}} - C_{\text{Verify}} \end{aligned} \tag{63}$$

where R^{SBM}_{Include} is the incentive of including verification confirmation details, containing metadata and performance index. The SBMs are responsible for verifying the performance of trained models, so they will be rewarded R_{Verify} according to the number of verified models, corresponding to the cost for receiving all encrypted models $C^{SBM}_{\text{RecFHEMs}}$ and verifying them C_{Verify}. C_{Mine} is the mining cost.

The SBM may not rank the trained models properly (IRa), so the strategy set of SBM is {Normal (N), Improper Rank (IRa)}. The utilities of different strategies are formulated as follows:

(6.1) SBM with N:

The revenue of SBM with N is given by:

$$\begin{aligned} R^{SBM} &= R^{\text{SBM}}_{\text{Include}} + R_{\text{Verify}} \\ &= Q_{\text{VerifiedM}} \cdot \mathcal{R}_{\text{VerifiedM}} + Q_{\text{VerifiedM}} \cdot Q_{\text{Cases}} \cdot \mathcal{R}_{\text{Verify}} \end{aligned} \tag{64}$$

where $Q_{\text{VerifiedM}}$ is the number of verified models, and $\mathcal{R}_{\text{VerifiedM}}$ is the corresponding incentive for including the records. Moreover, the system will incentivize the SBM to verify trained models based on the testing data provided by the previous TBM. Thus, there is a revenue for verification, where Q_{Cases} is the number of testing data cases, and $\mathcal{R}_{\text{Verify}}$ is the corresponding revenue for each case of verification per model.

The cost of SBM with N is given by:

$$\begin{aligned} C^{SBM} &= C_{\text{Mine}} + C^{SBM}_{\text{RecFHEMs}} + C_{\text{Verify}} \\ &= C_{\text{Mine}} + Q_{\text{VerifiedM}} \cdot k_{\text{Transmit}} \cdot k_{\text{Expand}} \cdot |M| \\ &\quad + Q_{\text{VerifiedM}} \cdot Q_{\text{Cases}} \cdot C^{\text{Unit}}_{\text{Verify}} \end{aligned} \tag{65}$$

where C_{Mine} is the mining cost. To verify the trained model, there is a receiving cost of encrypted models for the SBM, where k_{Transmit} is the transmission cost coefficient, k_{Expand} is the expanding coefficient of model parameter number after FHE, and $|M|$ denotes the number of parameters in the model. Moreover, the verification process has a computational cost, where $C^{\text{Unit}}_{\text{Verify}}$ denotes the verification cost per testing data case/unit.

Therefore, the utility of SBM with N can be formulated as:

$$\begin{aligned} U^{SBM}_{N} &= R^{SBM} - C^{SBM} \\ &= Q_{\text{VerifiedM}} \cdot \mathcal{R}_{\text{VerifiedM}} + Q_{\text{VerifiedM}} \cdot Q_{\text{Cases}} \cdot \mathcal{R}_{\text{Verify}} \\ &\quad - C_{\text{Mine}} - Q_{\text{VerifiedM}} \cdot k_{\text{Transmit}} \cdot k_{\text{Expand}} \cdot |M| \\ &\quad - Q_{\text{VerifiedM}} \cdot Q_{\text{Cases}} \cdot C^{\text{Unit}}_{\text{Verify}} \end{aligned} \tag{66}$$

(6.2) SBM with IRa:

The revenue of SBM with IRa is given by:

$$R^{SBM} = 0 \tag{67}$$

where the revenue of dishonest SBM will be equal to zero. This is because a dishonest SBM will be recognized by most participants, since everyone can check the correctness of the ranking results. Therefore, rational participants will build a new fork to invalidate the block from a dishonest SBM.

The cost of SBM with IRa is given by:

$$C^{SBM} = C_{\text{Mine}} \tag{68}$$

where the dishonest TBMs only have the mining cost C_{Mine}, since they will not truly receive or verify the trained models. If the TBM has honestly finished the verification process, there is no reason that a rational TBM improperly ranks the models due to the huge cost of the verification process.

Therefore, the utility of SBM with IRa can be calculated as:

$$\begin{aligned} U_{IRa}^{SBM} &= R^{SBM} - C^{SBM} \\ &= 0 - C_{\text{Mine}} \\ &= -C_{\text{Mine}} \end{aligned} \tag{69}$$

Overall, all utilities of different participants' strategies were formulated. Specifically, the final formulations of each strategy are summarized in Table II of Section III-C2.

B. Theoretical Analysis

1) Individual Rationality (IR): To achieve IR in the DeRelayL system, all participants that choose the "Normal" strategy should at least have positive utilities, which means that the incentive provided by the proposed mechanism should lead to $U_N^{Participant} \geq 0$. Therefore, for each participant, there will be some conditions to guarantee that $U_N^{Participant} \geq 0$, which can be presented as follows:

(1) IR for MO:

Let $U_N^{MO} \geq 0$, the IR condition for MO can be formulated as:

$$\frac{\overline{Q^{MO}_{\text{Selected}}} \cdot \mathcal{R}_{\text{Cited}}}{1-\beta} - Q_{\text{Selected}} \cdot (1-s) \cdot b^{MO} - k_{\text{Transmit}} \cdot |M| \geq 0 \tag{70}$$

Rearranging the inequality, the condition can be formulated as:

$$\frac{\overline{Q^{MO}_{\text{Selected}}} \cdot \mathcal{R}_{\text{Cited}}}{1-\beta} \geq Q_{\text{Selected}} \cdot (1-s) \cdot b^{MO} + k_{\text{Transmit}} \cdot |M| \tag{71}$$

Multiplying both sides by $(1-\beta)$ to eliminate the denominator:

$$\begin{aligned} &\overline{Q^{MO}_{\text{Selected}}} \cdot \mathcal{R}_{\text{Cited}} \geq \\ &\qquad (1-\beta) \cdot \big(Q_{\text{Selected}} \cdot (1-s) \cdot b^{MO} + k_{\text{Transmit}} \cdot |M|\big) \end{aligned} \tag{72}$$

Thus, to hold IR for MO, $\mathcal{R}_{\text{Cited}}$ should satisfy:

$$\mathcal{R}_{\text{Cited}} \geq \frac{(1-\beta) \cdot \big(Q_{\text{Selected}} \cdot (1-s) \cdot b^{MO} + k_{\text{Transmit}} \cdot |M|\big)}{\overline{Q^{MO}_{\text{Selected}}}} \tag{73}$$

(2) IR for T:

Let $U_N^T \geq 0$, the IR condition for T can be formulated as:

$$\begin{aligned} \big(V_{\text{RecM}} - V_{\text{Now}}^T + 1\big) \cdot \textcircled{C} + \frac{\overline{Q^{T}_{\text{Selected}}} \cdot \beta \cdot \mathcal{R}_{\text{Cited}}}{1-\beta} \\ \geq P_{\text{Comp}} \cdot D \cdot \tau \cdot |M| + (1-s) \cdot b^T \\ + k_{\text{Transmit}} \cdot |M| + k_{\text{Encrypt}} \cdot |M| \\ + Q_{\text{Broadcast}} \cdot k_{\text{Transmit}} \cdot k_{\text{Expand}} \cdot |M| \end{aligned} \tag{74}$$

Rearranging terms, the condition can be formulated as:

$$\begin{aligned} \frac{\overline{Q^{T}_{\text{Selected}}} \cdot \beta \cdot \mathcal{R}_{\text{Cited}}}{1-\beta} \geq P_{\text{Comp}} \cdot D \cdot \tau \cdot |M| + (1-s) \cdot b^T \\ + k_{\text{Transmit}} \cdot |M| + k_{\text{Encrypt}} \cdot |M| \\ + Q_{\text{Broadcast}} \cdot k_{\text{Transmit}} \cdot k_{\text{Expand}} \cdot |M| \\ - \big(V_{\text{RecM}} - V_{\text{Now}}^T + 1\big) \cdot \textcircled{C} \end{aligned} \tag{75}$$

Multiplying by $(1-\beta)$, the condition can be formulated as:

$$\begin{aligned} &\overline{Q^{T}_{\text{Selected}}} \cdot \beta \cdot \mathcal{R}_{\text{Cited}} \\ &\quad \geq (1-\beta) \cdot \big(P_{\text{Comp}} \cdot D \cdot \tau \cdot |M| + (1-s) \cdot b^T \\ &\qquad + k_{\text{Transmit}} \cdot |M| + k_{\text{Encrypt}} \cdot |M| \\ &\qquad + Q_{\text{Broadcast}} \cdot k_{\text{Transmit}} \cdot k_{\text{Expand}} \cdot |M| \\ &\qquad - \big(V_{\text{RecM}} - V_{\text{Now}}^T + 1\big) \cdot \textcircled{C}\,\big) \end{aligned} \tag{76}$$

Thus, to hold IR for T, $\mathcal{R}_{\text{Cited}}$ should satisfy:

$$\begin{aligned} \mathcal{R}_{\text{Cited}} \geq \frac{(1-\beta)}{\overline{Q^{T}_{\text{Selected}}} \cdot \beta} \cdot \Big(P_{\text{Comp}} \cdot D \cdot \tau \cdot |M| \\ + (1-s) \cdot b^T + k_{\text{Transmit}} \cdot |M| + k_{\text{Encrypt}} \cdot |M| \\ + Q_{\text{Broadcast}} \cdot k_{\text{Transmit}} \cdot k_{\text{Expand}} \cdot |M| \\ - \big(V_{\text{RecM}} - V_{\text{Now}}^T + 1\big) \cdot \textcircled{C}\Big) \end{aligned} \tag{77}$$

(3) IR for DBM:

Let $U_N^{DBM} \geq 0$, the IR condition for DBM can be formulated as:

$$U_N^{DBM} = Q_{\text{Deposit}} \cdot \mathcal{R}_{\text{Deposit}} - C_{Mine} \geq 0 \tag{78}$$

Therefore, to hold IR for DBM, $\mathcal{R}_{\text{Deposit}}$ should satisfy:

$$\mathcal{R}_{\text{Deposit}} \geq \frac{C_{\text{Mine}}}{Q_{\text{Deposit}}} \tag{79}$$

(4) IR for EBM:

Let $U_N^{EBM} \geq 0$, the IR condition for EBM can be formulated as:

$$\begin{aligned} Q_{\text{HashM}} \cdot \mathcal{R}_{\text{HashM}} + (V_{\text{FHEM}} - V_{\text{Now}}^{EBM}) \cdot \textcircled{C} \geq C_{\text{Mine}} \\ + k_{\text{Transmit}} \cdot k_{\text{Expand}} \cdot |M| + C_{\text{GenFHEKey}} \end{aligned} \tag{80}$$

Rearranging the terms, the condition can be formulated as:

$$\begin{aligned} Q_{\text{HashM}} \cdot \mathcal{R}_{\text{HashM}} \geq C_{\text{Mine}} + k_{\text{Transmit}} \cdot k_{\text{Expand}} \cdot |M| \\ + C_{\text{GenFHEKey}} - (V_{\text{FHEM}} - V_{\text{Now}}^{EBM}) \cdot \textcircled{C} \end{aligned} \tag{81}$$

Therefore, to hold IR for DBM, $\mathcal{R}_{\text{HashM}}$ should satisfy:

$$\begin{aligned} \mathcal{R}_{\text{HashM}} \geq \frac{1}{Q_{\text{HashM}}} \cdot \Big(C_{\text{Mine}} + k_{\text{Transmit}} \cdot k_{\text{Expand}} \cdot |M| \\ + C_{\text{GenFHEKey}} - (V_{\text{FHEM}} - V_{\text{Now}}^{EBM}) \cdot \textcircled{C}\Big) \end{aligned} \tag{82}$$

(5) IR for TBM:

Let $U_N^{TBM} \geq 0$, the IR condition for TBM can be formulated as:

$$\begin{aligned} Q_{\text{EncryptedM}} \cdot \mathcal{R}_{\text{EncryptedM}} + Q_{\text{Cases}} \cdot \mathcal{R}_{\text{Case}} \\ > C_{\text{Mine}} + Q_{Cases} \cdot C_{\text{GenTDCase}}^{\text{Unit}} \end{aligned} \tag{83}$$

where there are two parameters that should be determined to hold IR for TBM, including $\mathcal{R}_{\text{EncryptedM}}$ and $\mathcal{R}_{\text{Case}}$.

(6) IR for SBM:

Let $U_N^{SBM} \geq 0$, the IR condition for SBM can be formulated as:

$$\begin{aligned} Q_{\text{VerifiedM}} \cdot \mathcal{R}_{\text{VerifiedM}} + Q_{\text{VerifiedM}} \cdot Q_{\text{Cases}} \cdot \mathcal{R}_{\text{Verify}} \\ \geq C_{\text{Mine}} + Q_{\text{VerifiedM}} \cdot k_{\text{Transmit}} \cdot k_{\text{Expand}} \cdot \|M\| \\ + Q_{\text{VerifiedM}} \cdot Q_{\text{Cases}} \cdot C_{\text{Verify}}^{\text{Unit}} \end{aligned} \tag{84}$$

where there are two parameters that should be determined to hold IR for SBM, including $\mathcal{R}_{\text{VerifiedM}}$ and $\mathcal{R}_{\text{Verify}}$.

2) Incentive Compatibility (IC): To achieve IC in the DeRelayL system, all rational participants will tend to choose the "Normal" strategy, which means that the utility of the "Normal" strategy should be greater than other strategies. Therefore, for each participant, the incentive provided by the proposed mechanism should lead to $U_N^{Participant} \geq U_{OtherStrategy}^{Participant}$, which can be formulated as follows:

(1) IC for MO:

The strategy set of MO is $\{$Normal (N), Not Transmitting (NTm)$\}$. Therefore, we will compare the utility of MO with N and MO with NTm:

$$U_N^{MO} - U_{NTm}^{MO} = U_N^{MO} + Q_{\text{Selected}} \cdot b^{MO} > 0 \tag{85}$$

where $U_N^{MO} \geq 0$ due to the IR for EBM, and the revenue for including records $Q_{\text{Selected}} \cdot b^{MO} > 0$. Therefore, the utility of MO with N is greater than MO with NTm, ensuring IC for MO.

(2) IC for T:

The strategy set of T is $\{$Normal (N), Not Training (NTr), Not Broadcasting (NBr)$\}$. Therefore, we will first compare the utilities of T with N and T with NTr:

$$\begin{aligned} U_N^T - U_{NTr}^T &= U_N^T - (V_{\text{RecM}} - V_{\text{Now}}^T) \cdot \copyright \\ &\quad + b^T + k_{\text{Transmit}} \cdot |M| \\ &> -(V_{\text{RecM}} - V_{\text{Now}}^T) \cdot \copyright + b^T \\ &> 0 \end{aligned} \tag{86}$$

where $(V_{\text{RecM}} - V_{\text{Now}}^T) \cdot \copyright$ is the value of the version gap between the received model (the latest model) and the model that T has already owned (the out-of-date model). The formula $b^T - (V_{\text{RecM}} - V_{\text{Now}}^T) \cdot \copyright > 0$ means that the deposit of T should not be lower than the value of model (i.e., an effective deposit discussed in Section III-B). Otherwise, T will not have the motivation to train the model and just cheat for the latest models by depositing a small amount of coins. Therefore, the mechanism should have a condition:

$$b^T > (V_{\text{RecM}} - V_{\text{Now}}^T) \cdot \copyright \tag{87}$$

Then, we will compare the utilities of T with N and T with NBr:

$$\begin{aligned} &U_N^T - U_{NBr}^T = \\ &\quad \left(\left(V_{\text{RecM}} - V_{\text{Now}}^T + 1\right) \cdot \copyright + \frac{\overline{Q_{\text{Selected}}^T} \cdot \beta \cdot \mathcal{R}_{\text{Cited}}}{1 - \beta} \right) \\ &\quad - \left(P_{\text{Comp}} \cdot D \cdot \tau \cdot |M| + (1-s) \cdot b^T + k_{\text{Transmit}} \cdot |M| \right. \\ &\quad \left. + k_{\text{Encrypt}} \cdot |M| + Q_{\text{Broadcast}} \cdot k_{\text{Transmit}} \cdot k_{\text{Expand}} \cdot |M|\right) \\ &\quad - \left(\left(V_{\text{RecM}} - V_{\text{Now}}^T + 1\right) \cdot \copyright\right) \\ &\quad + \left(P_{\text{Comp}} \cdot D \cdot \tau \cdot |M| + b^T + k_{\text{Transmit}} \cdot |M|\right) \end{aligned} \tag{88}$$

Eliminating the common terms, the formula can be presented as:

$$\begin{aligned} U_N^T - U_{NBr}^T = \frac{\overline{Q_{\text{Selected}}^T} \cdot \beta \cdot \mathcal{R}_{\text{Cited}}}{1 - \beta} \\ - \left((-s) \cdot b^T + k_{\text{Encrypt}} \cdot |M|\right) \\ - (Q_{\text{Broadcast}} \cdot k_{\text{Transmit}} \cdot k_{\text{Expand}} \cdot |M|) \end{aligned} \tag{89}$$

Thus, to ensure $U_N^T - U_{NBr}^T > 0$, the mechanism should satisfy:

$$\begin{aligned} \frac{\overline{Q_{\text{Selected}}^T} \cdot \beta \cdot \mathcal{R}_{\text{Cited}}}{1 - \beta} > (-s) \cdot b^T + k_{\text{Encrypt}} \cdot |M| \\ + Q_{\text{Broadcast}} \cdot k_{\text{Transmit}} \cdot k_{\text{Expand}} \cdot |M| \end{aligned} \tag{90}$$

where the condition can be formulated as:

$$\begin{aligned} \mathcal{R}_{\text{Cited}} > \frac{1}{\overline{Q_{\text{Selected}}^T} \cdot \beta} \Big((-s) \cdot b^T + k_{\text{Encrypt}} \cdot |M| \\ + Q_{\text{Broadcast}} \cdot k_{\text{Transmit}} \cdot k_{\text{Expand}} \cdot |M|\Big) \cdot (1 - \beta) \end{aligned} \tag{91}$$

Therefore, there are two conditions (Formula (87) and Formula (91)) that should be satisfied in the mechanism design to ensure IC for T.

(3) IC for DBM:

The strategy set of DBM is $\{$Normal (N), Not Packing All (NPA), Packing Improper Deposit Contracts (PI)$\}$. Therefore, we will first compare the utilities of DBM with N and DBM with NPA:

$$\begin{aligned} U_N^{DBM} - U_{NPA}^{DBM} = (Q_{\text{Deposit}} \cdot \mathcal{R}_{\text{Deposit}} - C_{\text{Mine}}) \\ - (Q_{\text{DepositLess}} \cdot \mathcal{R}_{\text{Deposit}} - C_{\text{Mine}}) \end{aligned} \tag{92}$$

The expression can be simplified as:

$$U_N^{DBM} - U_{NPA}^{DBM} = (Q_{\text{Deposit}} - Q_{\text{DepositLess}}) \cdot \mathcal{R}_{\text{Deposit}} \tag{93}$$

Since $Q_{\text{Deposit}} > Q_{\text{DepositLess}}$, we can know:

$$U_N^{DBM} - U_{NPA}^{DBM} > 0 \tag{94}$$

Therefore, the utility of DBM with N is greater than DBM with NPA. On the other hand, we will first compare the utilities of DBM with N and DBM with PI:

$$\begin{aligned} U_N^{DBM} - U_{\text{PI}}^{DBM} = (Q_{\text{Deposit}} \cdot \mathcal{R}_{\text{Deposit}} - C_{\text{Mine}}) \\ - (-C_{\text{Mine}}) \end{aligned} \tag{95}$$

The expression can be simplified as:

$$U_N^{DBM} - U_{\text{PI}}^{DBM} = Q_{\text{Deposit}} \cdot \mathcal{R}_{\text{Deposit}} > 0 \tag{96}$$

Therefore, the utility of DBM with N is greater than DBM with PI. Overall, the IC for DBM can be ensured.

(4) IC for EBM:

The strategy set of EBM is {Normal (N), Not Generating FHE Key (NG)}. Therefore, we will compare the utilities of EBM with N and EBM with NG:

$$U_N^{EBM} - U_{NG}^{EBM} = U_N^{EBM} + C_{\text{Mine}} > 0 \tag{97}$$

where $U_N^{EBM} \geq 0$ due to the IR for EBM, and mining cost $C_{\text{Mine}} > 0$, so the $U_N^{EBM} - U_{NG}^{EBM} > 0$. Therefore, the utility of EBM with N is greater than EBM with NG, ensuring IC for EBM.

(5) IC for TBM:

The strategy set of TBM is {Normal (N), Improper Testing Cases (IT)}. Therefore, we will compare the utilities of TBM with N and TBM with IT:

$$U_N^{TBM} - U_{IT}^{TBM} = U_N^{TBM} + C_{\text{Mine}} > 0 \tag{98}$$

where $U_N^{TBM} \geq 0$ due to the IR for TBM, and mining cost $C_{\text{Mine}} > 0$, so the $U_N^{TBM} - U_{NG}^{TBM} > 0$. Therefore, the utility of TBM with N is greater than TBM with IT, ensuring IC for TBM.

(6) IC for SBM:

The strategy set of SBM is {Normal (N), Improper Rank (IRa)}. Therefore, we will compare the utilities of SBM with N and SBM with IRa:

$$U_N^{SBM} - U_{IRa}^{SBM} = U_N^{SBM} + C_{\text{Mine}} > 0 \tag{99}$$

where $U_N^{SBM} \geq 0$ due to the IR for TBM, and mining cost $C_{\text{Mine}} > 0$, so the $U_N^{SBM} - U_{IRa}^{SBM} > 0$. Therefore, the utility of SBM with N is greater than SBM with IRa, ensuring IC for SBM.

C. Overall Condition Set

Overall, according to the calculation in the previous subsections, the reward ($\mathcal{R}$) of each block should satisfy the following condition set (**T1 - T8**):

T1: To guarantee **IR** of MO, we need to let $U_N^{MO} \geq 0$, thus:

$$\mathcal{R}_{\text{Cited}} \geq \frac{(1-\beta) \cdot \big(Q_{\text{Selected}} \cdot (1-s) \cdot b^{MO} + k_{\text{Transmit}} \cdot |M|\big)}{\overline{Q_{\text{Selected}}^{MO}}} \tag{100}$$

T2: To guarantee **IR** of T, we need to let $U_N^T \geq 0$, that is:

$$\begin{aligned}\mathcal{R}_{\text{Cited}} \geq \frac{(1-\beta)}{\overline{Q_{\text{Selected}}^{T}} \cdot \beta} \cdot \Big(& P_{\text{Comp}} \cdot D \cdot \tau \cdot |M| \\ &+ (1-s) \cdot b^T + k_{\text{Transmit}} \cdot |M| + k_{\text{Encrypt}} \cdot |M| \\ &+ Q_{\text{Broadcast}} \cdot k_{\text{Transmit}} \cdot k_{\text{Expand}} \cdot |M| \\ &- \big(V_{\text{RecM}} - V_{\text{Now}}^T + 1\big) \cdot \copyright\Big)\end{aligned} \tag{101}$$

T3: To guarantee **IR** of DBM, we need to let $U_N^{DBM} \geq 0$, thus:

$$\mathcal{R}_{\text{Deposit}} \geq \frac{C_{\text{Mine}}}{Q_{\text{Deposit}}} \tag{102}$$

T4: To guarantee **IR** of EBM, we need to let $U_N^{EBM} \geq 0$, that is:

$$\begin{aligned}\mathcal{R}_{\text{HashM}} \geq \frac{1}{Q_{\text{HashM}}} \cdot \Big(& C_{\text{Mine}} + k_{\text{Transmit}} \cdot k_{\text{Expand}} \cdot |M| \\ &+ C_{\text{GenFHEKey}} - (V_{\text{FHEM}} - V_{\text{Now}}^{EBM}) \cdot \copyright\Big)\end{aligned} \tag{103}$$

T5: To guarantee **IR** of TBM, we need to let $U_N^{TBM} \geq 0$, thus:

$$\begin{aligned}Q_{\text{EncryptedM}} \cdot \mathcal{R}_{\text{EncryptedM}} + Q_{\text{Cases}} \cdot \mathcal{R}_{\text{Case}} \\ \geq C_{\text{Mine}} + Q_{Cases} \cdot C_{\text{GenTDCase}}^{\text{Unit}}\end{aligned} \tag{104}$$

T6: To guarantee **IR** of SBM, we need to let $U_N^{SBM} \geq 0$, that is:

$$\begin{aligned}Q_{\text{VerifiedM}} \cdot \mathcal{R}_{\text{VerifiedM}} + Q_{\text{VerifiedM}} \cdot Q_{\text{Cases}} \cdot \mathcal{R}_{\text{Verify}} \\ \geq C_{\text{Mine}} + Q_{\text{VerifiedM}} \cdot k_{\text{Transmit}} \cdot k_{\text{Expand}} \cdot |M| \\ + Q_{\text{VerifiedM}} \cdot Q_{\text{Cases}} \cdot C_{\text{Verify}}^{\text{Unit}}\end{aligned} \tag{105}$$

To satisfy **IC**, the utilities of the "Normal" strategy should be greater than other strategies. According to Table II, some participants' other strategies have negative utilities, so they will choose "Normal" obviously. Specifically, trainer T requires additional constraints for the strategy of "Not Training" and "Not Broadcasting":

T7: For **IC** of T with "Not Training", there is a sufficient but not necessary condition that the deposit of T should not be lower than the value of the model (i.e., an effective deposit discussed in Section III-B). Otherwise, T will not have the motivation to train the model.

$$b^T > (V_{\text{RecM}} - V_{\text{Now}}^T) \cdot \copyright \tag{106}$$

T8: For **IC** of T with "Not Broadcasting", let $U_N^T - U_{NBr}^T > 0$:

$$\begin{aligned}\mathcal{R}_{\text{Cited}} > \frac{1}{\overline{Q_{\text{Selected}}^{T}} \cdot \beta} \Big(& (-s) \cdot b^T + k_{\text{Encrypt}} \cdot |M| \\ &+ Q_{\text{Broadcast}} \cdot k_{\text{Transmit}} \cdot k_{\text{Expand}} \cdot |M|\Big) \cdot (1-\beta)\end{aligned} \tag{107}$$

The fore-mentioned condition set is listed in Section III-C2.