applied sciences

MDPI

Article

Blockchained Trustable Federated Learning Utilizing Voting Accountability for Malicious Actor Mitigation

Brian Stanley, Sang-Gon Lee * and Elizabeth Nathania Witanto

College of Software Convergence, Dongseo University, Busan 47011, Republic of Korea; briansta1601@gmail.com (B.S.); elizabethnathaniaw93@gmail.com (E.N.W.)
* Correspondence: nok60@dongseo.ac.kr

Abstract: The federated learning (FL) approach in machine learning preserves user privacy during data collection. However, traditional FL schemes still rely on a centralized server, making them vulnerable to security risks, such as data breaches and tampering of models caused by malicious actors attempting to gain access by masquerading as trainers. To address these issues that hamper the trustability of federated learning, requirements were analyzed for several of these problems. The findings revealed that issues, such as the lack of accountability management, malicious actor mitigation, and model leakage, remained unaddressed in prior works. To fill this gap, a blockchain-based trustable FL scheme, MAM-FL, is proposed with the focus on providing accountability to trainers. MAM-FL established a group of voters responsible for evaluating and verifying the validity of the model updates submitted. The effectiveness of MAM-FL was tested based on the reduction of malicious actors present on both trainers' and voters' sides and the ability to handle colluding participants. Experiments show that MAM-FL succeeded at reducing the number of malicious actors, despite the test case involving initial collusion in the system.

Keywords: federated learning; blockchain; trustability

Citation: Stanley, B.; Lee, S.-G.; Witanto, E.N. Blockchained Trustable Federated Learning Utilizing Voting Accountability for Malicious Actor Mitigation. *Appl. Sci.* **2023**, *13*, 6707. https://doi.org/10.3390/app13116707

Academic Editor: Yoshiyasu Takefuji

Received: 28 April 2023
Revised: 23 May 2023
Accepted: 24 May 2023
Published: 31 May 2023

1. Introduction

Federated learning (FL) is a method of preserving user privacy during data collection in machine learning. However, traditional FL schemes rely on a centralized server to aggregate model updates, making them vulnerable to security risks, such as data breaches, unauthorized access, and tampering with models [1]. An example of a data breach is the act of eavesdropping on communication channels during weight and gradient updates [2,3]. The effect of such an act can cause loss of privacy, reduced public trust in FL systems, and legal and financial liabilities. Additionally, Sybil attacks, where an attacker creates multiple fake identities, manipulates FL results, and consumes resources, can also hurt the trustability of FL in general. Another set of problems is the reliance on a single central server, which is vulnerable to denial-of-service attacks as the system expands [4]; moreover, there is a concern regarding the emergence of bad faith actors who may destabilize the system as more people become involved [1].

To counteract the vulnerabilities in traditional FL, as mentioned above, replacing the server with a blockchain as an aggregator has been a solution utilized in multiple works [5–16]. References [5,11,12] added incentive mechanisms that encouraged honest parties. References [6–8] introduced generic designs for efficient and private FL systems that protect against inference attacks on clients' model updates by using secure aggregation. References [10,13–16] focused on blockchain infrastructure. These studies leveraged smart contracts, improved consensus algorithms to evaluate participant contributions, and implemented incentive mechanisms within the framework. One common aspect among the mentioned works is the absence of accountability management. During training, trainers might misbehave, and the system cannot differentiate whether a trainer is fully trustworthy

until the end. For example, a hacker might gain access to a trainer's account and use its identity to inject bad updates to the model, while masquerading as a normal, uncompromised account. There is also the unmentioned problem of model leakage during transmission, which can allow attackers to access the data by eavesdropping on the communication channels, or masquerading as a participant in the scheme.

In order to tackle the problems, such as the lack of accountability management, malicious actor prevention, and model leakage, MAM-FL, a scheme that addresses vulnerabilities, is proposed. First, in MAM-FL, signcryption is adopted to prevent model and privacy leakage by providing encryption, signature authentication, and integrity verification. An accountability-checking mechanism is integrated through a party of voters to verify the accuracy claims of trainers. Adding a verifying party will create a need for trainers to assume responsibility, which provides accountability and ensures consistency in the quality of the model updates. MAM-FL consists of an initial setup, leading up to the creation of the next global model, in which the selected model updates are chosen through the result of ranking the scores in the model updates. Throughout MAM-FL, blockchain and the InterPlanetary File System (IPFS) [17] act as communication channels to provide reliability and resistance toward single-point-of-failure (SPoF) caused by DDoS attacks and spam, which is a property of a decentralized system. Finally, to motivate participants to join and then perform honestly, a reward system is proposed, which grows proportionately based on the participant's initial deposit.

The following contributions are provided in this paper:

1. We address several concerns of traditional federated learning and formulate seven requirements based on existing solutions. We provide reasoning to explain how MAM-FL can fix the problems mentioned in the seven requirements.
2. Signcryption is integrated into the initial model transmission to provide confidentiality; blockchain provides reliable protection against single points of failure and verifies the integrity of data during communication.
3. An FL protocol utilizing a voting-based mechanism is proposed. The voting system is combined with a dynamic reputation system that affects the weight of the votes. The purpose is to introduce accountability measures that can detect and prevent malicious actors from jeopardizing the FL ecosystem.

The rest of this paper is organized as follows. First, the definitions are listed in Section 2. In Section 3, seven proposed requirements are shown, along with the state-of-the-art blockchain-based FL protocols. MAM-FL protocols are described in Section 4, and the whole scheme is evaluated in Section 5. Finally, Section 6 concludes the whole paper.

2. Preliminaries

2.1. Federated Learning

Federated learning is a machine learning technique that enables the training of models across multiple decentralized devices or servers, without transferring data to a central location [18,19]. In federated learning, the data remain on local devices, and the model is trained locally, with only the updated model parameters being sent to a central server for aggregation. This approach creates robust and scalable models while maintaining data privacy and security, as the raw data never leave the local device. Federated learning finds applications in scenarios where data are both sensitive and decentralized.

2.2. Blockchain

A blockchain is a decentralized and distributed digital ledger technology that enables secure and transparent record-keeping of transactions across a network of computers. A blockchain consists of a series of blocks containing a previous block's cryptographic hash, a timestamp, and transaction data. To provide a tamper-proof and immutable record of all transactions, once a block is added to the chain, it cannot be altered [20].

A single-point-of-failure is a component or node that, if it fails, causes the entire system to stop functioning. This is a concern in many centralized systems. A blockchain avoids

this problem by replicating the data and transactions across multiple nodes that form a peer-to-peer network. Even if some nodes are offline, corrupted, or malicious, it can still operate, as long as the majority of nodes are honest and reachable [21].

2.3. IPFS

IPFS is a distributed and decentralized peer-to-peer file-sharing system that aims to create a permanent and decentralized method of storing and sharing files on the internet [17]. IPFS stores files based on their content instead of their location, using a global network of interconnected nodes for faster and more efficient content distribution.

One of the key benefits of IPFS is that it allows for decentralized and censorship-resistant file sharing, making it useful for applications, such as content distribution and file archiving. It can also be used as a building block for decentralized applications, providing a secure and reliable method of storing and sharing data in a distributed network.

2.4. ZSS Signcryption

Reference [22] proposed a signcryption scheme, as depicted in Algorithm 1, which incorporated an improved version of a signature algorithm by Zhang, Safavi-Naini, and Susilo (ZSS). In prior signcryption work by [23], the user forwarded the original message m to a trusted third party (TTP). By doing so, user data might have leaked to the TTP and were no longer private. In the ZSS signature generation, the author included a random value r. The user also needed to pass the r value to the TTP, which allowed the TTP to compute a symmetric encryption key, k, by using $H_3(r)$, making it dangerous. Different from [23], the author suggests that in the process of verifying a message, the original message itself does not need to be sent to the verifier. This ensures that the message remains private and improves the efficiency of the verification process, as the verifier does not have to download the message beforehand.

Algorithm 1 Modified ZSS signcryption by [22]

Signcryption: When a user acts as the sender S, they generate a signcryption σ for the message m, as follows:

1. Generate $v = (Hash_1(m) + SK_S)^{-1}$.
2. Choose $r \leftarrow^R \{0,1\}^d$ and generate $V = vP$, $X = r \oplus Hash_2(V, PK_R, vPK_R)$. Then, generate $k = Hash_3(r)$, which is a symmetric encryption key. So, $Y = Enc_k(m)$.
3. Therefore, the signcryption $\sigma = (V, X, Y)$, where V is a ZSS signature of message m, X is the randomness of the encryption key generation, and Y is the encryption of the message m.

Unsigncryption: After receiving σ = (V, X, Y) from the sender S, the recipient R starts the unsigncryption process.

1. R parses σ to obtain (V, X, Y).
2. Compute $r = X \oplus Hash_2(V, PK_R, VSK_R)$ and $k = Hash_3(r)$.
3. Decrypt Y to obtain message m. So, $m = Dec_k(Y)$.
4. If Equation (1) holds, unsigncryption succeeds; otherwise, R rejects σ from the sender.

$$e(Hash_1(m)P + PK_S, V) = e(P, P) \quad (1)$$

- **Setup Phase**
  - **ParamGen.** Given the security parameters b and d, let G_1 be a cyclic additive group and G_2 be a multiplicative cyclic group with the prime order p; P is the generator of group G_1. The bilinear mapping e: $G_1 \times G_2 \rightarrow G_2$, three hash functions $Hash_1$: $\{0,1\}^* \rightarrow Z_p$, $Hash_2$: $G_1^3 \rightarrow \{0,1\}^d$, $Hash_3$: $\{0,1\}^d \rightarrow \{0,1\}^b$, and a symmetric encryption scheme (Enc, Dec) are utilized. Therefore, the system parameters are $\{b, d, G_1, G_2, P, e, Hash_1, Hash_2, Hash_3, Enc, Dec\}$.

– **KeyGen.** The sender chooses a random number from Z_p, sets it as the secret key SK_S, and computes the public key $PK_S = SK_S P$. The recipient chooses a random number from Z_p, sets it as the secret key SK_R, and computes the public key $PK_R = SK_R P$.

3. Requirements for Trustability in Federated Learning

First, multiple requirements to satisfy trustability in a federated learning setting are defined. In each requirement, a discussion is made on how the property has partially been met by existing solutions previously researched.

R1 Confidentiality.

Malicious entities can attempt to obtain access to local models and global models by listening to communication channels and then modifying the model data; this involves changing the original owner's data in the model, claiming the model, and then using it for their own gain. Therefore, this requires a solution that can prevent the model from being inferred by the adversaries. One particular solution involves performing encryption [24] during the process of aggregating updates and on the original model itself, with a public key encryption scheme, in such a way that outside entities cannot recognize the contents of the data exchange.

R2 Attractiveness.

In FL, the organizer needs to provide a method for consistently recruiting new trainers, as without enough trainers, the resulting global model will not be accurate due to the limited amount of data. Furthermore, having more trainers will increase data diversity, as they will train with different datasets based on their own devices. However, from a trainer's perspective, engaging in local training imposes a cost on their resources. The workers will most likely not perform the FL tasks in the vanilla FL without the motivation to perform the training. To counter this, one author proposed an incentive system to encourage trainers to join the training, mostly via token-based rewards [25].

R3 Accountability.

The major problem with a centralized FL model is that it lacks mechanisms to hold each participant accountable in a fair and objective way [26]. Some trainers can perform training using low-quality datasets on their devices, which can impact the quality of the expected model, and the absence of accountability can be a problem, as this will cause a continuous loop of bad updates from the trainers. The system needs to prepare a solution that can encourage trainers to train with better data while also being capable of detecting and punishing trainers who intentionally manipulate their model accuracy. One author's solution is to impose a reputation system with levels [27], combined with the incentive system that rewards trainers based on their reputations. This encourages trainers to exhibit positive behavior and improve their reputations. If a trainer continues to perform honest training without any detected misbehavior, they will receive increased rewards at the end of the training, while dishonest ones are punished. By imposing this rule, trainers cannot easily remove themselves or reduce their participation, since doing so would diminish the tokens they deposited; this responsibility would compel trainers to continue through the process with good behavior.

R4 Reliability.

As the number of trainers increases, a single central server might not have enough capability to handle the communication overhead. Attackers can attempt DDoS (distributed denial-of-service) attacks, which will generate significant overhead for the central server, causing the whole FL system to crash and shut down in the process. The authors of Reference [28] suggested a committee of nodes that could serve as the replacement for the central aggregator. This committee is selected based on a metric system that takes into account factors such as liveliness (the rate at which a node remains available and is not down) and reputation. The most trustworthy nodes will be more likely to be selected as committee members.

R5 Consistency.

The lack of trainers with stable internet connectivity might hamper the ability to upload complete parts of their updates since an unreliable network cannot guarantee the full transfer of the update parameters. Malicious trainers can also use this method of dropping and rejoining the model training to intentionally submit low-quality updates. All of this will cause the global model to be updated with broken chunks of updates, hampering accuracy. One proposed scheme [27] imposes a time limit to prevent low-connectivity trainers from sending broken or corrupt updates to the server. This solution attempts to prevent trainers with low connectivity from submitting their updates, which can harm the model's accuracy because the broken parts are being submitted instead of the whole parts.

R6 Integrity

Model integrity is critical in FL because the accuracy and effectiveness of the machine learning model depend on the quality and reliability of the data used for training. Any tampering or corruption of the data can lead to inaccurate or biased results, which can impact the performance of the model and its ability to make accurate predictions. In an FL environment, data are distributed across multiple devices or clients, and updates are made to the model by aggregating the results of the training performed on each device. As a result, it is essential to ensure that the data used for training are complete, accurate, and consistent across all devices. Blockchain technology can be used as an integrity solution for the federated learning training process by providing a transparent and tamper-proof ledger of all transactions and computations performed during the training process [20]. By using a blockchain, participants in the federated learning network can record their contributions to the training process, including the models they trained and the data they used. These contributions can be verified by the other participants in the network by using the blockchain, ensuring that all participants are contributing in a fair and transparent manner.

R7 Authentication

Authentication is critical in FL because it ensures that only authorized devices or clients participate in the training process and make updates to the machine learning model. Without authentication, there is a risk of unauthorized access to the data and the model, which can lead to data breaches, theft of intellectual property, and other security issues. In an FL environment, authentication is typically achieved through secure communication protocols, such as transport layer security (TLS) and secure sockets layer (SSL) [29]. These protocols encrypt the data being transmitted between devices or clients, ensuring that they cannot be intercepted or modified by unauthorized parties. Authentication can also be achieved through the use of digital certificates and public key cryptography. Each device or client can be issued a digital certificate that contains a public key, which is used to verify the identity of the device or client during the authentication process. The device or client can then use the private key to sign the updates it makes to the model, ensuring that they are authentic and cannot be tampered with.

Previous Approaches and Limitations

Several blockchain-based frameworks and protocols, such as BFL [5], BEAS [6], ByToChain [8], DeTrustFL [9], ModelChain [13], Twin FL [14], and SEC-CL [15] lack incentive mechanisms in their systems, which may cause participants to have no motivation to share their data or model updates with the model's owner, leading to low participation rates and poor learning performances. Meanwhile, DeepChain [12] offers blockchain-based FL with incentive- and deposit-based systems; however, it does not integrate a reputation system, which could serve as a vulnerability as attackers can masquerade as trainers.

References [5,16] utilized practical Byzantine fault tolerance (PBFT) as an alternative to PoW consensus during training. However, PBFT is limited in that the system cannot tolerate a number of malicious nodes in the network equal to or greater than one-third of the total number of nodes. While the reputation system can enhance the reliability of

the system to some extent, there is a lack of external verification of the reputation, such as through a third-party review process.

Reference [7] proposed a blockchain-based FL framework with differential privacy as encryption. Differential privacy helps protect the privacy of individuals in a dataset by adding random noise to the data or the queries. This noise introduces a trade-off; while providing privacy, it can affect the utility or accuracy of the data analysis [30].

Reference [10] proposed a permissioned blockchain-based federated learning method, where incremental updates to an anomaly detection machine learning model are chained together on the distributed ledger. However, this paper focused more on the machine learning model and did not mention much about the importance of incentive mechanisms and other aspects, such as security, accountability, or confidentiality of FL.

Reference [11] leveraged blockchain-sharding features to enable collaborative model training in a more distributed and trustworthy manner. Blockchain sharding, however, can compromise security and decentralization, as each shard may have fewer nodes and less hash power than the whole network, making it more vulnerable to attacks or collusion.

Reference [27] proposed a blockchain-based scheme with a leveling system, wherein trainers gain experience and reputation over time, leading to better rewards for their performance. However, since the proposed peer-review system forces each trainer to also act as a reviewer, it can be deemed unreliable due to the necessity of multitasking [31].

The research conducted by [32] centered on the integration of FL in a vehicular ad hoc network (VANET) environment. The study assessed the performance of devices, the efficacy of ML and aggregation algorithms, the effects on edge-to-server communication, and resource consumption. However, it did not address the incentive mechanism, confidentiality, and reliability aspects essential for sustaining federated learning.

Reference [33] introduced the incorporation of local differential privacy and zero-knowledge proof in a blockchain-based FL framework. While this work provides confidentiality using differential privacy and incentives, it does not provide a trust management system to combat malicious actors masquerading as honest workers.

Based on these observations, Table 1 shows which references satisfied the requirements listed in Section 3, marked by ✓, while × is for the unsatisfied ones.

Table 1. List of works and which requirements are fulfilled.

Reference	Confidentiality	Attractiveness	Accountability	Reliability	Consistency	Integrity	Authentication
[5]	×	×	✓	✓	×	×	×
[6]	✓	×	✓	×	×	×	×
[7]	✓	✓	✓	✓	×	×	×
[8]	×	×	✓	×	×	×	×
[9]	×	×	✓	×	×	×	×
[10]	×	×	✓	×	×	×	×
[11]	×	×	×	✓	✓	×	×
[12]	×	✓	×	×	×	×	×
[13]	✓	×	×	×	×	×	×
[14]	✓	×	×	×	×	×	×
[15]	×	×	✓	×	×	×	×
[16]	×	✓	×	×	×	×	×
[26]	✓	×	×	×	×	×	×
[27]	✓	✓	✓	✓	✓	×	×
[28]	✓	×	×	×	✓	×	×
[32]	×	×	×	×	✓	×	×
[33]	×	×	×	✓	✓	×	×
[34]	×	✓	✓	✓	×	×	×
MAM-FL	✓	✓	✓	✓	✓	✓	✓

To improve accountability, MAM-FL utilizes a voting-based system to select models for averaging. In the beginning, trainers must first deposit a specific amount of tokens in

order to join the network and participate in the training and voting process. This deposit system serves as an added layer of security, as it ensures that clients have a vested interest in the network and are less likely to act maliciously. The added voting mechanism imposes responsibility to both trainers and voters; an honest voter will have a more significant impact on the overall score of the model updates that will be accepted, while a malicious voter can be identified for collaborating with other malicious trainers, and both will be punished together, accordingly.

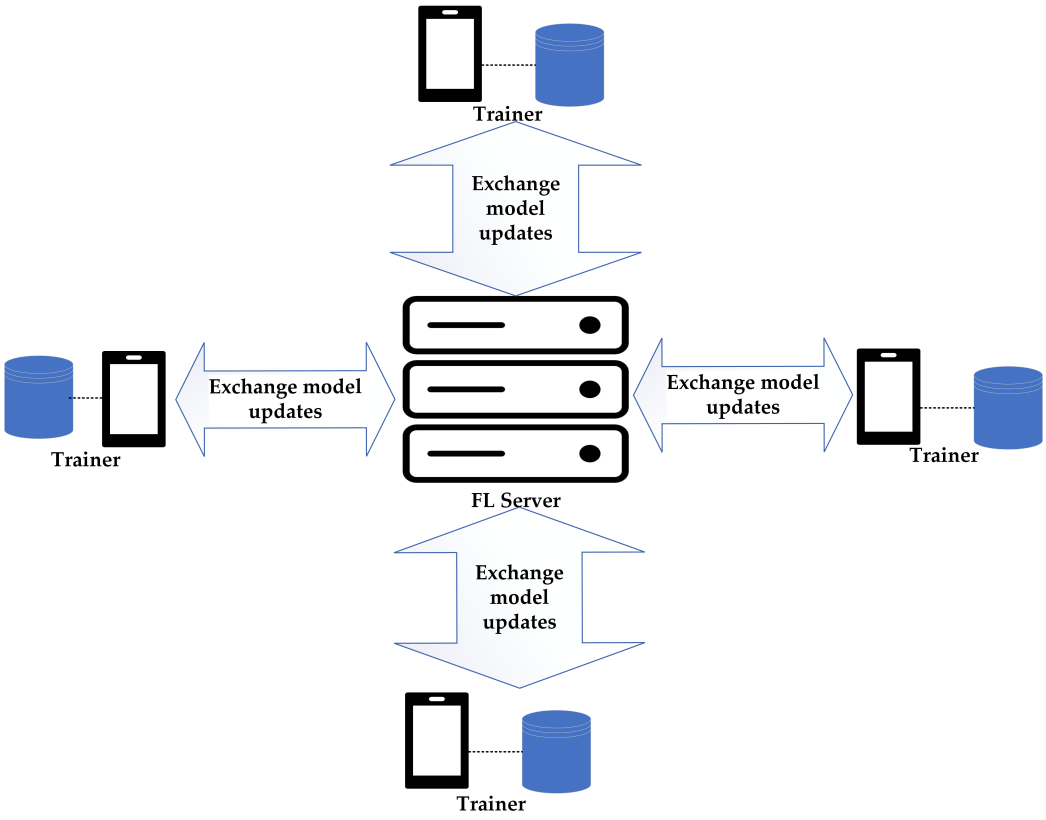
In addition to the voting and deposit systems, MAM-FL adds a reputation system for participants. This reputation system will track participants' past behaviors, such as their participation and contribution to the network, as well as any negative actions they may have taken. Clients with higher reputations will more likely be selected for model averaging and receive higher rewards.

Some aspects of importance that previous works have yet to highlight are integrity and authentication. Blockchain has been used as a method to provide incentive systems and FL reliability (e.g., SPoF) but prior studies have not highlighted the immutability of the blockchain as a key feature [5–8,12,13,16,33]. Leveraging the blockchain as a verification mechanism can provide a way to ensure integrity of the data transfer process in federated learning.

4. Proposed Protocol

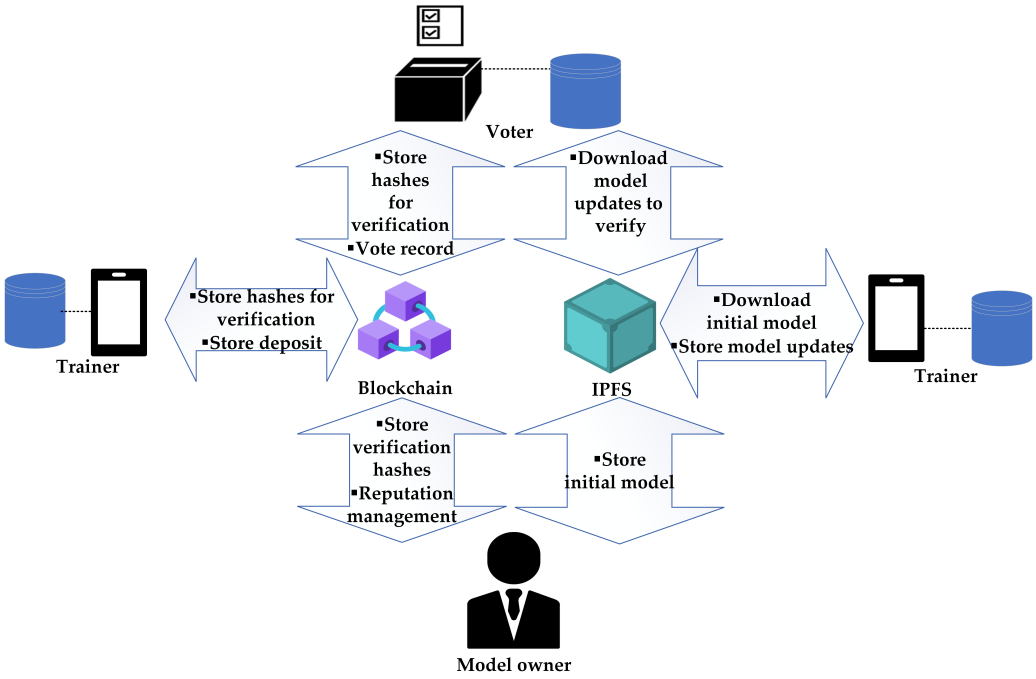
4.1. Comparison to Traditional FL

The traditional FL scheme depicted in Figure 1a is composed of a central server located in a cloud-based environment, and trainers interacting with the central server. When a component of the FL system fails, the entire system's performance is affected. An attacker can exploit this weakness by targeting the central server, preventing the scheme from being executed when the server goes down.



(**a**) Traditional FL Scheme

Figure 1. *Cont.*



(**b**) MAM-FL

Figure 1. Above is (**a**) the traditional FL scheme compared to (**b**) MAM-FL.

Figure 1a shows the relationship diagram of a common FL scheme, where trainers interact directly with the central server, while Figure 1b, MAM-FL, where trainers and voters interact with the blockchain and decentralized storage to perform FL training. The conventional FL approach is prone to SPoF as it relies on a central server to manage communication between trainers. This increases the risk of models being compromised during data aggregation through the communication endpoint. To address this issue, MAM-FL is proposed, as shown in Figure 1b. MAM-FL incorporates blockchain technology to mitigate the vulnerability of the centralized server. The immutable nature of the blockchain also enables it to function as a verification mechanism by storing hashes of the initial model and subsequent updates. However, due to the typically large sizes of models, storing them in the blockchain can be expensive and resource-intensive. To mitigate this challenge, a decentralized storage solution is added, which avoids the SPoF that would exist if a central storage solution were used instead. Overall, the MAM-FL architecture provides several enhancements over the traditional FL approach, including a reputation and voting system that ensures accountability and prevents malicious actors from compromising the FL process.

4.2. Proposed Protocol Details

Figure 2 shows the overall process of MAM-FL. The protocol is composed of interactions between trainers, voters, processing devices, and storage devices. Overall, the protocol restarts at the end of the sixth stage, with subsequent iterations making use of the average model in the fifth stage (federated averaging).

Trainers and voters are free to join the FL scheme by first depositing their initial token numbers. Each will be assigned an initial reputation with equal values. The number of tokens deposited does not amount to special benefits regarding the participant's role but will increase the rewards the trainer will gain at the end of each iteration. The list of notations used in this paper is available in Table 2.

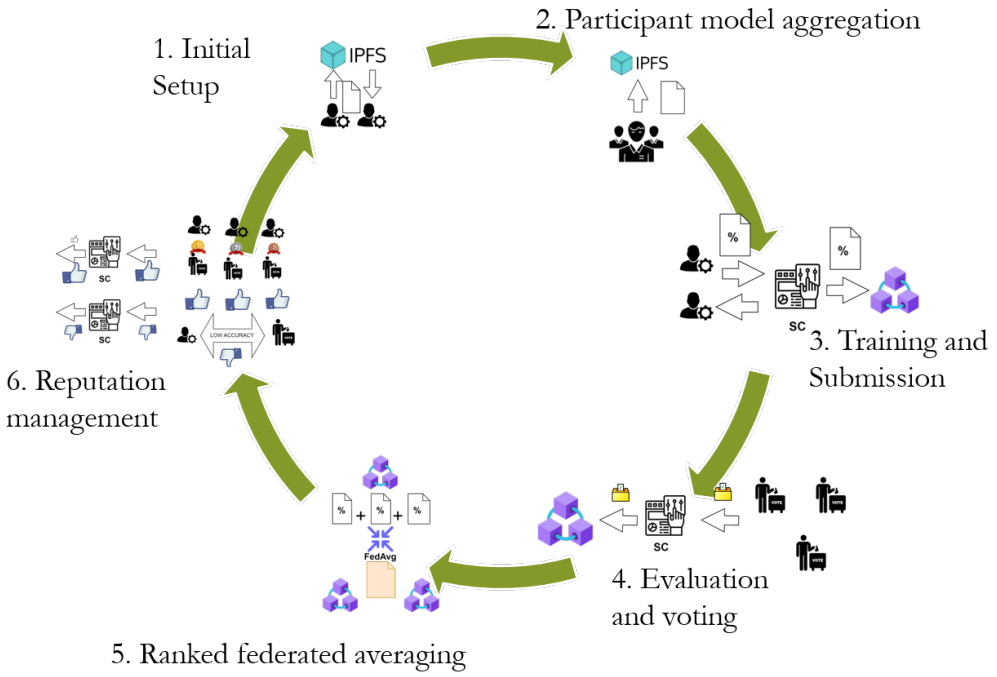


Figure 2. The overall proposed FL process.

Table 2. List of notations used in this section.

Notation	Description
M_0	Initial global model created by the owner
M_u	Trainer's generated model update
MU	Array of M_u
t	Time limit for trainers to submit the model updates
x	Minimum token deposit
r	Minimum reputation threshold
α	Consists of M_0, t, x, and r
δ	The accuracy claimed by the trainers
β	Consists of M_u and δ
γ	Voters' decisions, can be *accept/reject*
Ω	Resulting model from federated averaging
V	ZSS signature of the message m
X	Randomness of the encryption key generation
Y	Signcryption message (the result of signcryption)
H	Hash function
SC	Smart contract

4.3. Initial Setup

The initial setup process is depicted in Figure 3; it begins with the global owner of the model performing signcryption on the model and then submitting it to IPFS. This setup process ends with the owner submitting the IPFS hash of the encrypted model and signcryption parameters to the smart contract. By utilizing the blockchain's immutable property, the model data can be prevented from being compromised by malicious actors. The details of the process are explained as follows.

1. The owner of the model prepares the initial model α, consisting of $M_0 \mid t \mid x \mid r$. They then perform ZSS signcryption (as in Algorithm 1) to generate $(V_\alpha, X_\alpha, Y_\alpha)$. In this case, V_α is the signature of α, X_α is the randomness of the key, and Y_α is the encrypted version of α. Starting from the second iteration, the initial model will use the new global model, which is the result of the previous iteration's final model.
2. The owner sends the Y_α to IPFS.

3. IPFS stores Y_α.
4. IPFS generates a unique cryptographic hash for the uploaded model (α_{ipfs} and returns it to the owner.
5. The owner submits X, V, α_{ipfs} in the SC.
6. The SC stores X, V, α_{ipfs}.
7. The SC relays the successful transaction event to the owner of the model.

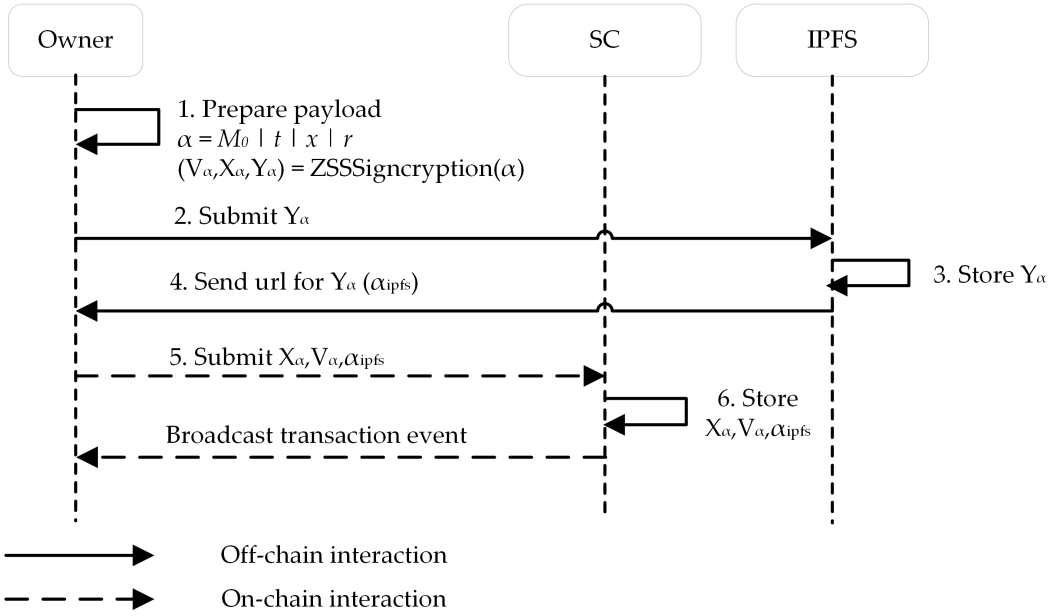


Figure 3. The sequence diagram for the initial setup.

4.4. Participant Model Aggregation

After the initial setup, participants download the model from the IPFS. The participants first need to request the initial model's location URL to the smart contract. The process, as depicted in Figure 4, is conducted by each participant (both trainers and voters) after the aggregated model is uploaded to IPFS. The details of the process are explained in the next passage.

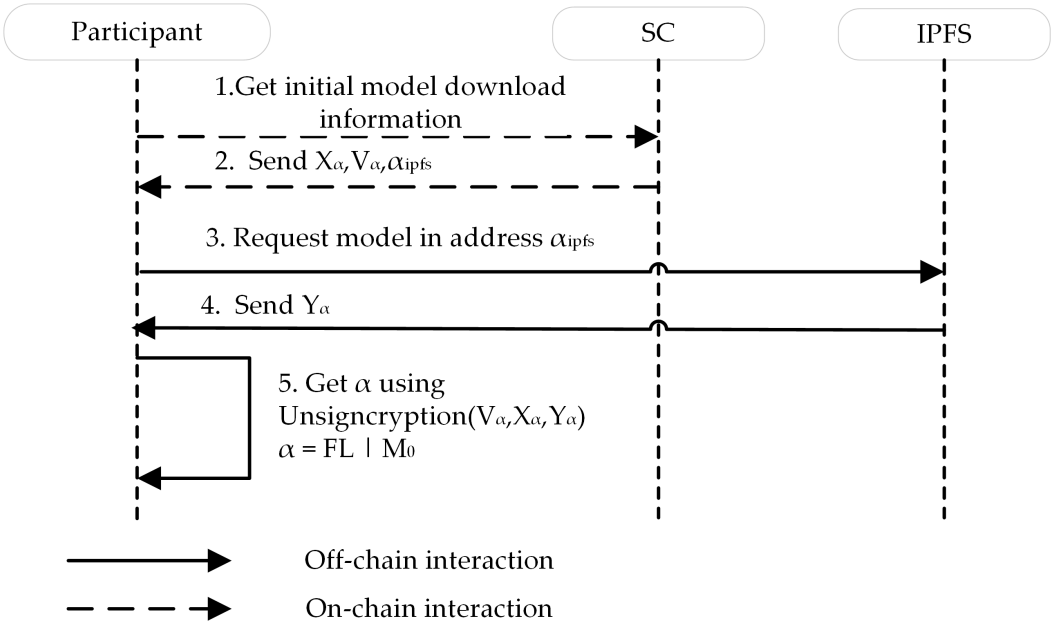


Figure 4. The sequence diagram for participant model aggregation.

1. The participant requests the initial model URL to the *SC*.
2. The *SC* sends X_α, V_α, and α_{ipfs} to the participant.
3. The participant requests Y_α to IPFS based on the URL provided in α_{ipfs}.
4. IPFS returns Y_α to the participant.
5. The trainer performs unsigncryption (as illustrated in Algorithm 1) using $(V_\alpha, X_\alpha, Y_\alpha)$ to obtain α.

4.5. Training and Submission

During the training phase shown in Figure 5, the trainers perform training on the initial models on their devices. When the time limit of the training process is reached, the trainers claim their accuracies and aggregate their model updates for the current epoch. The process ends after each trainer submits the IPFS hash of their update and the model integrity hash to the smart contract for verification purposes. Further details are listed as follows.

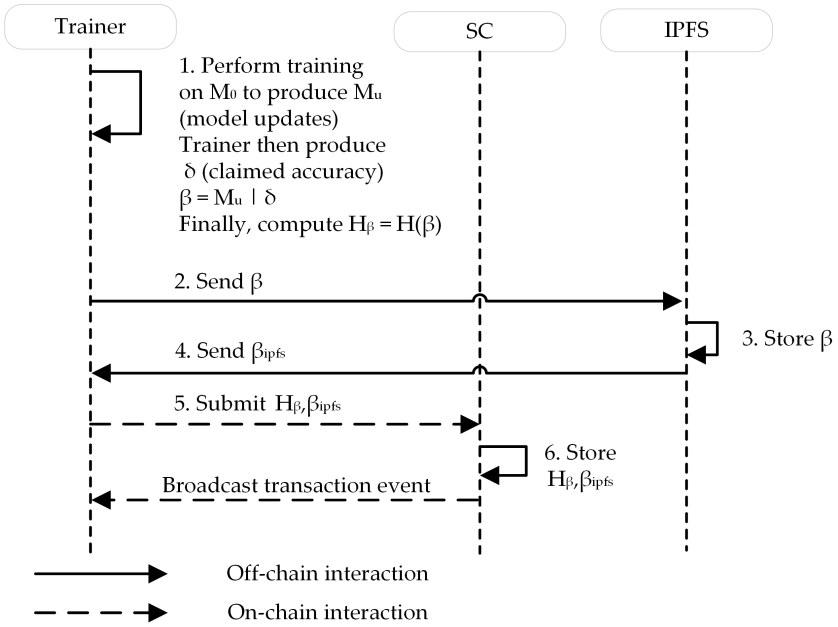


Figure 5. The sequence diagram for each trainer's training process.

1. The trainer performs model training on M_0 to generate β and δ (an accuracy claim based on the result). Honest trainers make truthful claims, whereas malicious actors may fabricate the accuracy of results. The trainer then creates H_β, which is the hash of β used to verify the integrity of the payload's content by the voter.
2. The trainer sends β to IPFS.
3. IPFS stores β.
4. IPFS generates a unique cryptographic hash for the trainer's model update (β_{ipfs}) and returns it to the trainer.
5. The trainer sends H_β and β_{ipfs} to the *SC*.
6. The *SC* stores H_β and β_{ipfs}.
7. The *SC* relays a successful transaction event to the trainer.

4.6. Evaluation and Voting

During the voting phase, as shown in Figure 6, voters request model update information from the smart contract for evaluation. In the end, the voters submit their votes to the smart contract. Details of the process are listed below.

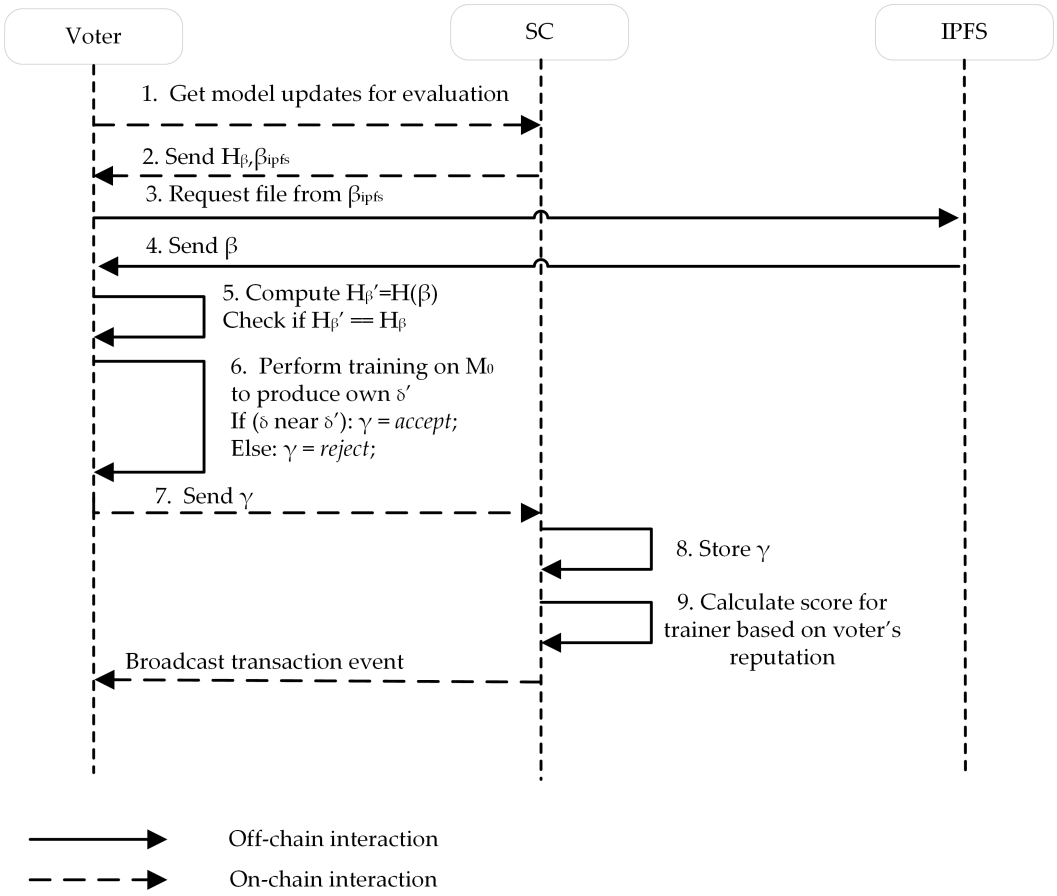


Figure 6. The sequence diagram of the voting procedure.

1. The voter requests the URL for model updates from the *SC*.
2. The *SC* sends H_β and β_{ipfs} to the voter.
3. The voter requests Y_β from the IPFS URL provided in β_{ipfs}.
4. IPFS returns the trainer's β to the voter.
5. IPFS computes $H_\beta{}'$, the hash of β, and compares the value with the downloaded H_β from the *SC* to verify the integrity of β.
6. The voter decrypts Y_α (as shown in Figure 4) and performs training on the decrypted M_0. They claim δ' and compare their scores with δ, producing γ (the decision of the voter). If the values are sufficiently close, the voter must choose **'accept'**; otherwise, the voter may choose **'reject'**. This step tests the honesty of voters, as dishonest voters may choose to *accept* even when accuracy fluctuates while rejecting the model updates that should have been accepted.
7. The voter sends γ to the *SC*.
8. The *SC* records γ.
9. The *SC* calculates the score of the trainer based on the current voter's reputation. For example, if a trainer has a score of x and a voter has a reputation of y, the current score is $x' = x + y$. Note that if the vote is a *'reject'* decision, the value of y is negative.
10. The *SC* relays a successful transaction event to the voter.

4.7. Ranked Federated Averaging

The Federated Averaging process is shown in Figure 7. The server will group and then tally the points of the model updates based on the number of **'accept'** and **'reject'** votes, resulting in S_T, which reflects the overall score of the model update from the respective trainer t. After the score tally is calculated, the T-highest scoring model updates will

be selected for federated averaging.). At the end of the process, the owner will apply punishment to lower-scoring trainers and voters who voted for the punished trainers.

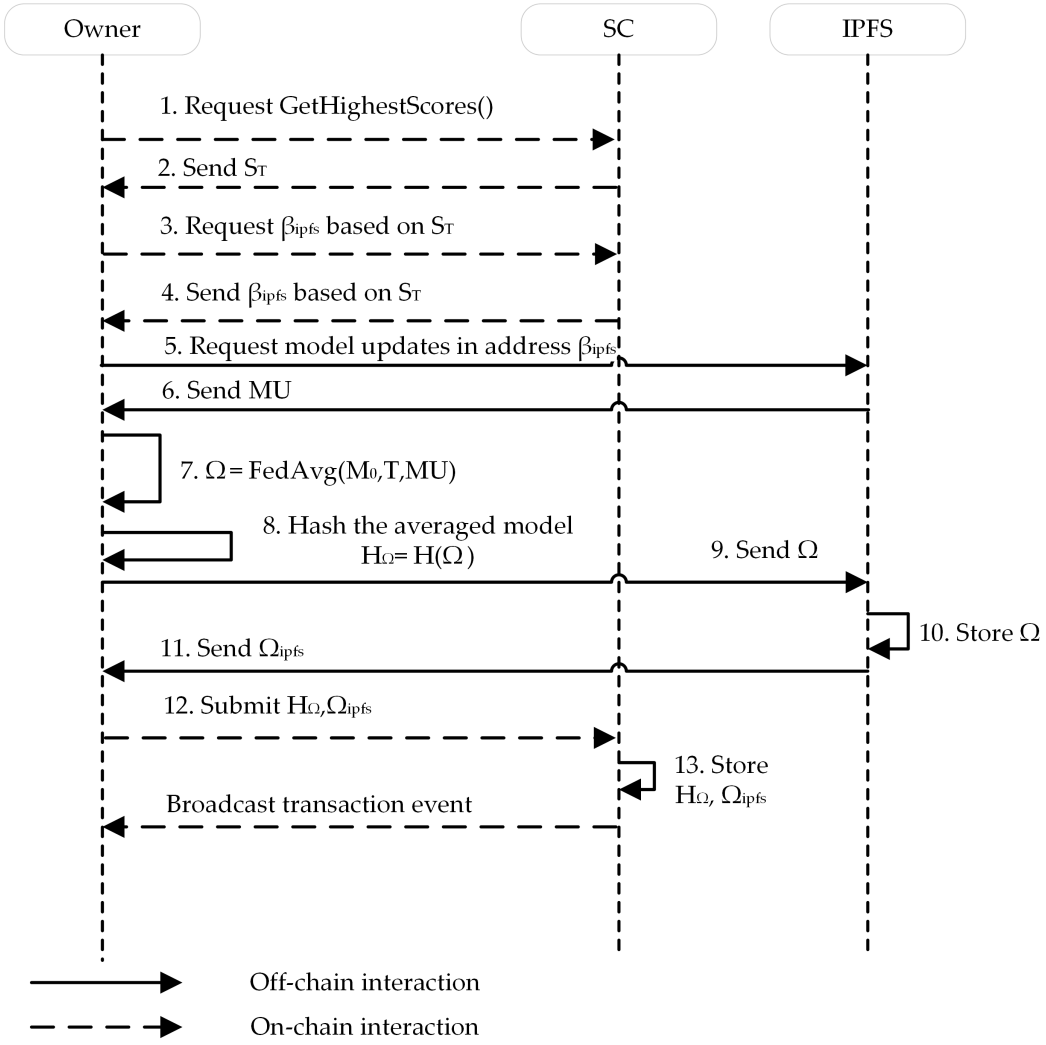


Figure 7. The sequence diagram for the federated averaging process.

1. The owner requests the result of the GetHighestScores() method (as shown in Algorithm 2) to the *SC* to obtain the highest scores of trainers based on the tally of accept and reject votes.
2. The *SC* invokes the GetHighestScores() method and returns S_T (the array of trainer IDs with the highest scores) to the owner. The value of T is equal to $N/2$ which represents half of the total amount of trainers, so only the top half are selected.
3. The owner requests an array of β_{ipfs} to the *SC* based on the trainer IDs in S_T.
4. The *SC* returns $\beta_{ipfs} = \{\beta_{ipfs1}, \beta_{ipfs2}, ..., \beta_{ipfsT}\}$ to the owner.
5. For each IPFS address, the owner requests respective model updates to IPFS based on the selected β_{ipfs} addresses.
6. IPFS returns $MU = \{MU_1, MU_2, ..., MU_T\}$ to the owner.
7. The owner then produces Ω, the resulting model from averaging the model updates (based on Algorithm 3).
8. The owner runs a hash function on the averaged model, producing H_Ω.
9. The owner sends Ω to IPFS.
10. IPFS stores Ω.
11. IPFS generates Ω_{ipfs}, a unique cryptographic hash for Ω, and returns it to the trainer.
12. The owner submits H_Ω, Ω_{ipfs} to the *SC*.

13. The *SC* stores H_Ω, Ω_{ipfs}.
14. The *SC* relays the successful transaction event to the owner of the model.

Algorithm 2 Get Highest Scores

```
function GETHIGHESTSCORES
    N ← (number of trainers)
    S ← (array of trainer scores)
    S′ ← sortDescending(S) ▷ sort the array in descending order from the highest value first, provided as a function in the smart contract
    S_T ← slice(S′,0,N/2) ▷ returns a section of an array, starting from the index specified in the second argument and ending at the index specified in the last argument
    return S_T
end function
```

Algorithm 3 Federated Averaging

```
function FEDAVG
    M_0 ← (initial global model)
    T ← (total amount of selected model updates)
    MU ← (array of model updates)
    Ω ← (M_0/T) Σ_{t=1}^{T} MU_t
    return Ω
end function
```

4.8. Reputation Management

The model owner incorporates reputation management in the sixth and final stage to evaluate participants' performances. Based on their reputations and current deposits, participants are rewarded accordingly. The procedural details are in the ADJUSTREPUTATION() function in Algorithm 4. In the ADJUSTREPUTATION() algorithm, first, it refers to the GETPUNISHLIST() function depicted in Algorithm 5 to obtain a list of participants who will receive a reputation penalty. Consider a scenario where there are N number of trainers with the highest scores, and the highest scores are considered $(T_1, T_2..., T_{N-1})$. Trainers whose scores are below T_{N-1} will be added to the punish list. For voters, the punished ones are those who have voted for trainers who were added to the punish list.

Algorithm 4 Reputation Management

```
function ADJUSTREPUTATION
    PT, PV ← GETPUNISHLIST()
    for each trainer t = 1, 2, ..., T do
        if t ∈ PT then
            penalize t's reputation
        end if
        reward t
    end for
    for each voter v = 1, 2, ..., V do
        if v ∈ PV then
            penalize v's reputation
        end if
        reward v
    end for
end function
```

Algorithm 5 Punish List

function GETPUNISHLIST
  $PT \leftarrow$ (initialize the empty array of punished trainers)
  $PV \leftarrow$ (initialize the empty array of punished voters)
  for each trainer $t = 1, 2, \ldots, T$ **do**
    if $S_t <$ lowest of S_T **then**
      add t to PT
    end if
  end for
  for each voter $v = 1, 2, \ldots, V$ **do**
    $Z_v \leftarrow$ (obtain the list of trainers that v voted *accept* for)
    if $Z_v \cap PT$ **then** ▷ Check if v voted for a punished trainer
      add v to PV
    end if
  end for
  return PT, PV
end function

5. Experimental and Proposed Scheme Analysis

The experiment was conducted on a machine with Windows 10 as the operating system with an Intel Core(TM) i7-8700K CPU and 32 GB RAM .A Docker container with 2 CPU cores and 2 GB of RAM was utilized to run Ganache. The smart contract was written in the Solidity language and deployed to Ganache using Truffle.

In the experiment, 10 iterations with 30 participants of FL simulations were performed, with the population composed as follows:

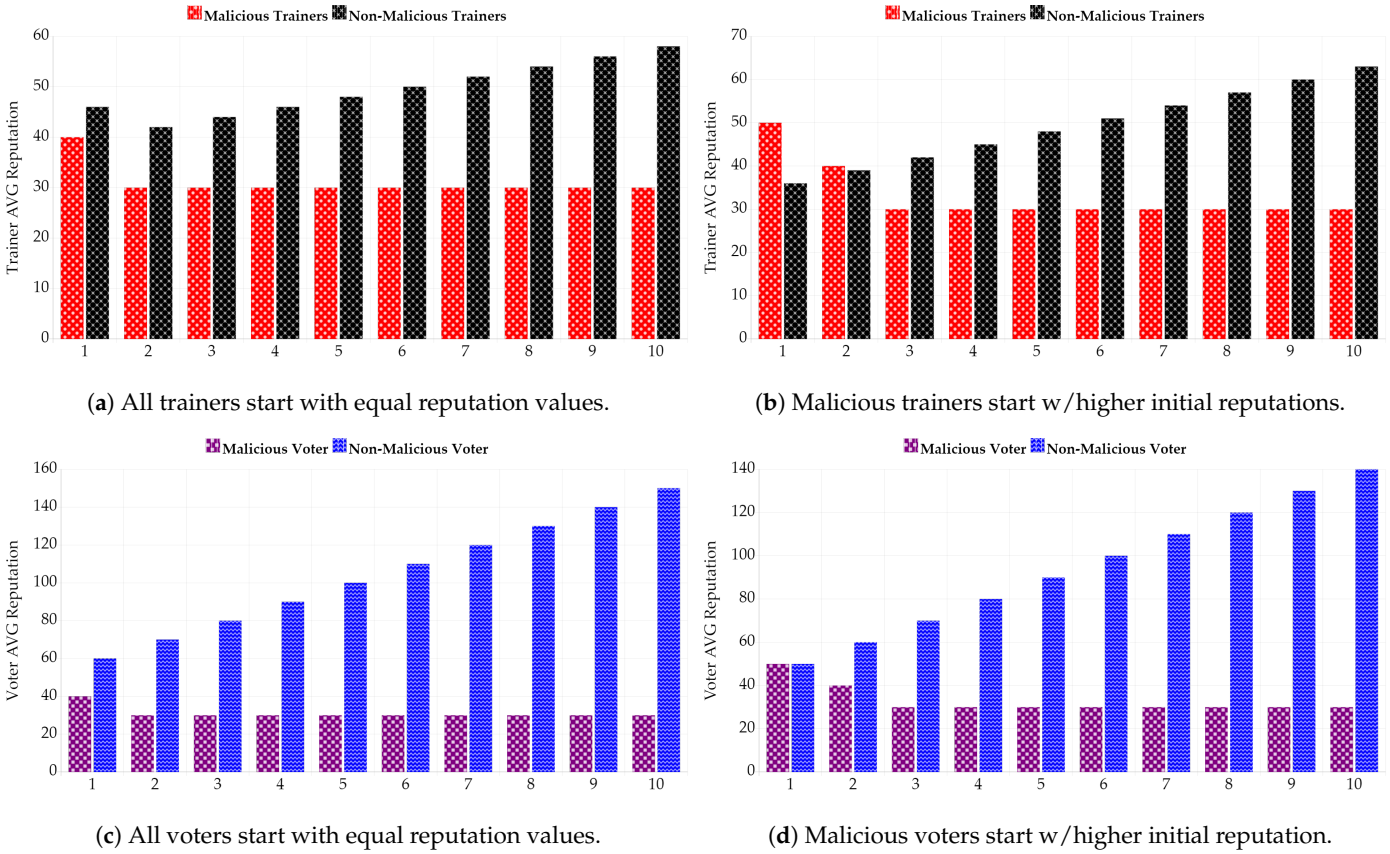
- 20 trainers (10 malicious, 10 non-malicious)
- 10 voters (3 malicious, 7 non-malicious)

To participate, trainers and voters must initially stake a minimum of 5 million gwei (which is equal to approximately ETH 0.005, which, as of 28 April 2023, was equal to USD 9.58), and will earn rewards that are proportional to their reputations. For example, a participant with x million gwei and y reputation will receive $(y/10)$% of their current gwei deposit, resulting in a new deposit of $x + (x * y/10)$ gwei for the next iteration. Therefore, maintaining consistent performance and reputation is crucial for participants to maximize their rewards.

5.1. Reputation Progression

Figure 8a shows that when the same reputations are established for all participants, malicious trainers are immediately banned from participating in their second iteration. There is also a slight decrease in the non-malicious trainer reputation in the second iteration, indicating some honest trainers can still submit low accuracies due to unforeseen circumstances, such as low bandwidth or unstable connectivity. A different parameter test is conducted with malicious trainers, starting with 60 initial reputation values (as opposed to honest ones starting with 40), as depicted in the results in Figure 8b. The rationale behind this test was to simulate a scenario where collusion had occurred, and the majority of the malicious actors had managed to obtain a higher reputation than the non-malicious ones. Even though the initial collusion might be successful, as reflected by the higher reputation scores, the model update ranking allows for the collusion to be detected over time. Figure 8c,d similarly depicts malicious voters that have a similar reputation progression to their trainer counterparts. Additionally, the reputation system is designed to identify trainers who submit low-quality updates, typically caused by connectivity issues. Such issues result in submitting broken update chunks that may lead to lower-than-expected accuracy. To ensure model consistency, the system is engineered to select only trainers with stable performance. During the evaluation phase in Section 4.6, voters detect low-accuracy updates and vote to reject them. The sum of votes with the accept, reject, and negative

votes reflects the results of the experiments, demonstrating that the system effectively filters out inconsistent trainers and malicious actors from the FL scheme.



(**a**) All trainers start with equal reputation values.

(**b**) Malicious trainers start w/higher initial reputations.

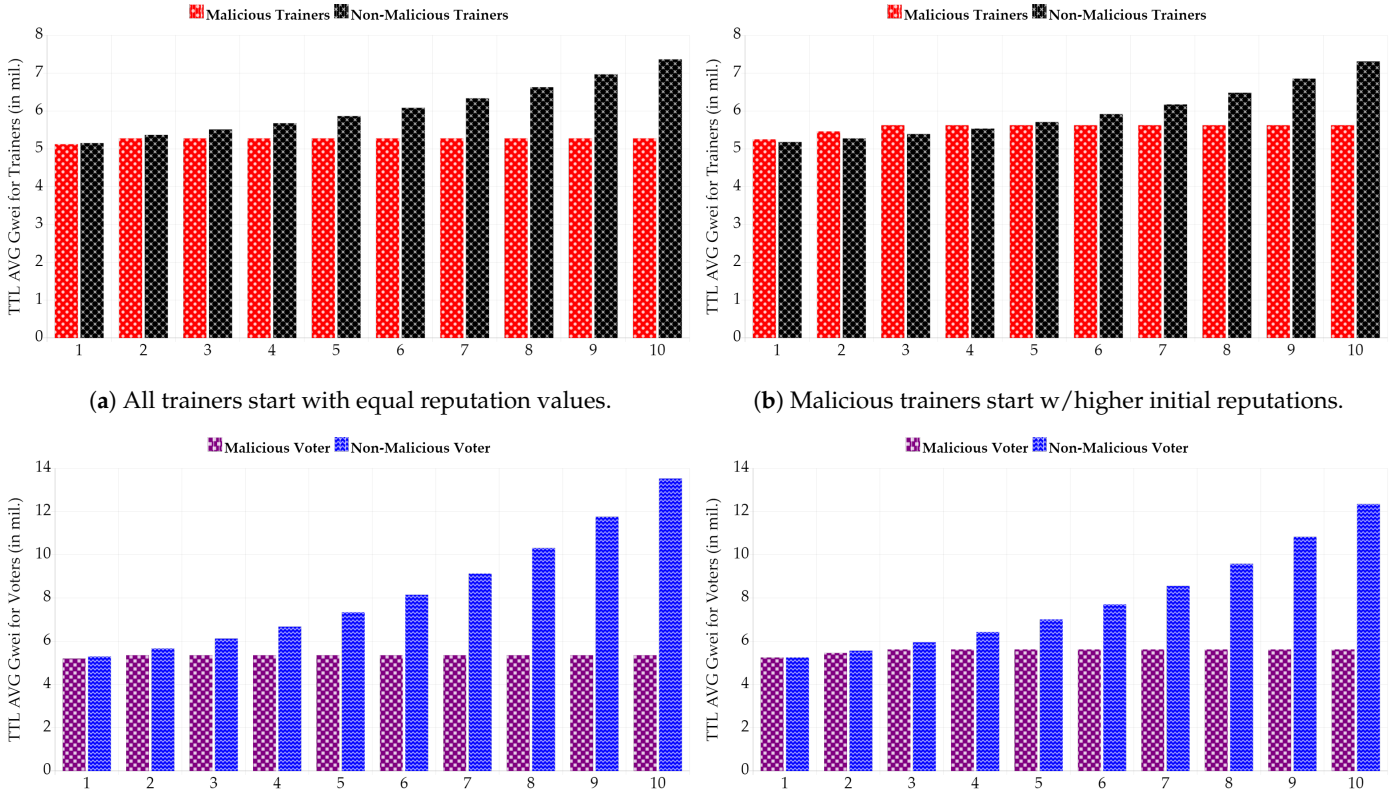
(**c**) All voters start with equal reputation values.

(**d**) Malicious voters start w/higher initial reputation.

Figure 8. The above two figures depict the average reputation values of malicious vs. non-malicious trainers at the end of each iteration, (**a**) starting with equal reputation values, compared to (**b**) malicious trainers having higher starting values. The below two figures depict the average reputation values of malicious vs. non-malicious voters at the end of each iteration, (**c**) starting with equal reputation values, compared to (**d**) malicious voters having higher starting values.

5.2. Reward Gain

The correlation between reward and reputation (Figures 8a–d and 9a–d) shows that maintaining reputation is essential to preventing an account ban. As shown in the ten iterations, participants with higher reputation values experience exponential increases in rewards. Compared to the first test with equal reputation values, the malicious participants' rewards start higher than those of honest ones. However, since the malicious trainer-submitted models are low in quality, their scores are lower than average and are subject to punishment. This results in a decrease in their reputation and leads to them being banned. When participants are banned, their accounts are frozen, temporarily preventing their deposited tokens from being withdrawn. This measure discourages participants from creating multiple accounts, as participants would need to stake tokens again to join. The reward system incentivizes trainers and voters to participate honestly in the schemes, as their rewards directly correlate with their reputations. Moreover, participants have a responsibility to maintain the tokens that they deposited at the start.



(**a**) All trainers start with equal reputation values.

(**b**) Malicious trainers start w/higher initial reputations.

(**c**) All voters start with equal reputation values.

(**d**) Malicious voters start w/higher initial reputations.

Figure 9. The above two figures depict the average reward values of malicious vs. non-malicious trainers at the end of each iteration, (**a**) starting with equal reputation values compared to (**b**) malicious trainers having higher starting values. The below two figures depict the average reputation values of malicious vs. non-malicious voters at the end of each iteration, (**c**) starting with equal reputation values compared to (**d**) malicious voters having higher starting values.

5.3. Gas Fee

The following tests were conducted using the Truffle testing library in a local Ganache environment. It has been verified that all of the implemented methods comply with the Ethereum gas limit standard of 30 million per block [35], rendering them feasible for execution on Ethereum networks. The gas usage values reported in Table 3 are calculated per unit, with each unit corresponding to a single instance of the method being executed. Each method serves a specific function in the smart contract, such as adding a new participant or updating global accuracy for the next iteration. For instance, "calculateReputation" calculates a participant's new reputation at the end of an iteration, and "setReputation" is used to update the value in the chain. It is important to note that the total number of transactions required for each method in a federated learning task depends heavily on the number of participating clients.

Table 3. The table displays the gas usage of methods used in the experiment.

Gas Fee Metrics		
Methods	**Gas Usage per Tx**	**% Limit**
addParticipant	301.020	1.003
calculateReputation	154.796	0.516
setGlobalAccuracy	26.646	0.089
setReputation	27.366	0.091

5.4. Requirement Comparisons Compared to Previous Works

Table 1 in Section 3 details the extent to which previous works have fulfilled the requirements. Table 4 shows the property comparison between the proposed scheme and other existing works. The benefits of MAM-FL are directly compared with two selected sources, namely References [7,27]. These papers were chosen because they meet most of the requirements compared to the other references.

Table 4. Comparison between works which requirements have fulfilled the most.

Reference	Confidentiality	Attractiveness	Accountability	Reliability	Consistency	Integrity	Authentication
[7]	Differential privacy	Deposit and reputation-based reward	Reputation system	Blockchain and IPFS	-	-	-
[27]	Standard encryption	Deposit and reputation-based reward	Reputation system	Blockchain and IPFS	Stage timeout	-	-
MAM-FL	Signcryption	Deposit and reputation-based reward	Voting and reputation	Blockchain and IPFS	Accuracy-based ranking	Blockchain hash and signcryption	Signature in signcryption

Confidentiality: Reference [7] uses differential privacy to maintain the confidentiality of data in federated learning. Differential privacy has an advantage over encryption in that it allows for data sharing and analysis for scientific or social purposes, while protecting the privacy of individual data entries. However, this technique may introduce data errors or uncertainty and conflict with other privacy technologies. Reference [27] adds a standard model encryption in the FL on the participant keys. While encryption can prevent unauthorized access to data and preserve the exact values, it may limit data usage and analysis and expose the data to re-identification attacks. MAM-FL uses a signcryption scheme to encrypt the initial model, which combines the functionalities of digital signatures and encryptions in a single operation. This is a reasonable trade-off since signcryption provides other requirements that are not available in standard encryption, such as authentication and integrity.

Attractiveness: References [7,27] and MAM-FL use rewards to serve as motivation for participants to maintain good behavior and high performance, ensuring the quality and effectiveness of the federated learning process.

Accountability: Both [27] and MAM-FL require participants to deposit a certain amount of tokens to participate, which creates an additional level of responsibility for the participants to continue the process until they receive their rewards. The risk of losing the stored tokens also adds protection against Sibyl attacks, hindering attackers who create multiple accounts. In Reference [27], a peer-review system was added, where each trainer acted as a reviewer for the other. However, this system can be unreliable since the trainers have to multitask and may lose focus. In contrast, MAM-FL assigns the roles of verifiers to voters, who contribute to the ranking of accepted model updates. This further helps establish accountability for each participant in the process without overwhelming them with too many tasks.

Reliability: References [7,27] and MAM-FL use blockchain, a distributed ledger, to record FL transactions. By nature, blockchain can address SPoF by allowing the participants to maintain a distributed ledger of the FL process, where each block records the model's updates and other relevant information. However, MAM-FL goes beyond simply integrating the process of each exchange between transmissions by utilizing smart contracts. Because storing model files can be costly, References [7,27] and MAM-FL utilize decentralized storage, with IPFS being chosen in this case.

Consistency: Reference [7] uses differential privacy, where the added noise can make it difficult to achieve convergence of the model across all clients. This is because the noise added to each client's data can be different, which can lead to inconsistencies in the model's updates. Reference [27] uses stage timeout to limit low-bandwidth trainers from submitting broken updates. In MAM-FL, a mechanism is included for tracking the reputation of each trainer based on their performance. Trainers with low reputations may be excluded from the process, ensuring that only high-performing trainers contribute to the model. This helps to protect the consistency of the scheme by weeding out low-performing trainers.

Integrity: In References [7,27], there is a lack of mention regarding integrity in FL, which can render the scheme vulnerable to attackers attempting to steal the model and perform modifications. In contrast, MAM-FL provides two methods to preserve integrity. First, to verify the signature of the encrypted model, Equation (1) requires a hash to be constructed, which will generate a different value if the model has been modified. Second, throughout the process, hashes of the model updates are verified to ensure the integrity of the model updates. The hashes are stored in the blockchain, while the model updates are stored in IPFS, which are then compared to guarantee that the contents are not modified. The added verification prevents data breaches since in the scenario of the stolen model, the contents can be compared to determine if the model is still genuine or not, preserving integrity.

Authentication: In References [7,27], the importance of authentication in FL is not mentioned. This introduces a vulnerability, where a malicious party could sabotage the aggregator by distributing fake versions of the model. The validity of the model cannot be determined. To solve such a problem, the MAM-FL uses signcryption to verify the legitimacy of the model, confirming whether the model is sent by the real owner or not.

6. Conclusions and Future Works

This research paper examines the requirements for establishing trust in a federated learning scheme, and identifies seven crucial factors, i.e., confidentiality, attractiveness, accountability, reliability, consistency, integrity, and authentication. The paper proposes MAM-FL, an FL scheme to address the requirements. First, the protocol integrates signcryption to encrypt the initial model to ensure the model's confidentiality, integrity, and origin authentication. Then, to enhance attractiveness, trainers and voters are guaranteed rewards based on their performance. Next, model updates are ranked and scored based on voter reputation, to address the accountability and consistency in the model updates. Lastly, the FL scheme is integrated with blockchain and IPFS to address single-point-of-failure, providing reliability.

To evaluate the effectiveness of the proposed protocol, experiments were conducted to reduce the number of malicious actors and incentivize honest contributions through a reputation-based reward system. The results demonstrate that the proposed protocol successfully mitigates the impact of malicious actors by using voter credibility to determine their trustworthiness, and the reputation-aware reward system is an attractive incentive for participants. The protocol was also tested in a scenario where malicious participants started with higher initial reputations; it still effectively reduced the number of malicious participants with subsequent iterations.

Future research can focus on measuring the response time of a blockchain-integrated FL scheme with a voting mechanism to determine its scalability. Additionally, since the paper defines that voters vote on all trainers, it can be useful to develop a worker selection algorithm to determine the assignment pairings of trainers and voters.

Author Contributions: Conceptualization, B.S.; data curation, B.S.; formal analysis, B.S.; funding acquisition, S.-G.L.; investigation, B.S.; methodology, B.S.; project administration, S.-G.L.; resources, S.-G.L.; supervision, S.-G.L.; validation, B.S. and S.-G.L.; visualization, B.S.; Writing—original draft, B.S.; writing—review and editing, B.S., S.-G.L. and E.N.W. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (grant number: 2018R1D1A1B07047601).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Kairouz, P.; McMahan, H.B.; Avent, B.; Bellet, A.; Bennis, M.; Bhagoji, A.N.; Bonawitz, K.; Charles, Z.; Cormode, G.; Cummings, R.; et al. Advances and Open Problems in Federated Learning. *arXiv* **2021**, arXiv:1912.04977.
2. Fredrikson, M.; Jha, S.; Ristenpart, T. Model Inversion Attacks that Exploit Confidence Information and Basic Countermeasures. In Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, CCS '15, New York, NY, USA, 12–16 October 2015; pp. 1322–1333. [CrossRef]
3. Yu, H.; Yang, S.; Zhu, S. Parallel Restarted SGD with Faster Convergence and Less Communication: Demystifying Why Model Averaging Works for Deep Learning. *arXiv* **2018**, arXiv:1807.06629.
4. Doriguzzi-Corin, R.; Siracusa, D. FLAD: Adaptive federated learning for DDoS attack detection. *arXiv* **2022**, arXiv:2205.06661.
5. Li, Y.; Chen, C.; Liu, N.; Huang, H.; Zheng, Z.; Yan, Q. A Blockchain-based Decentralized Federated Learning Framework with Committee Consensus. *IEEE Netw.* **2021**, *35*, 234–241. [CrossRef]
6. Mondal, A.; Virk, H.; Gupta, D. BEAS: Blockchain Enabled Asynchronous & Secure Federated Machine Learning. *arXiv* **2022**, arXiv:2202.02817.
7. Zhao, Y.; Zhao, J.; Jiang, L.; Tan, R.; Niyato, D.; Li, Z.; Lyu, L.; Liu, Y. Privacy-Preserving Blockchain-Based Federated Learning for IoT Devices. *arXiv* **2021**, arXiv:1906.10893.
8. Li, Z.; Yu, H.; Zhou, T.; Luo, L.; Fan, M.; Xu, Z.; Sun, G. Byzantine Resistant Secure Blockchained Federated Learning at the Edge. *IEEE Netw.* **2021**, *35*, 295–301. [CrossRef]
9. Xu, R.; Baracaldo, N.; Zhou, Y.; Anwar, A.; Kadhe, S.; Ludwig, H. DeTrust-FL: Privacy-Preserving Federated Learning in Decentralized Trust Setting. *arXiv* **2022**, arXiv:2207.07779.
10. Preuveneers, D.; Rimmer, V.; Tsingenopoulos, I.; Spooren, J.; Joosen, W.; Ilie-Zudor, E. Chained anomaly detection models for federated learning: An intrusion detection case study. *Appl. Sci.* **2018**, *8*, 2663. [CrossRef]
11. Moudoud, H.; Cherkaoui, S.; Khoukhi, L. Towards a Secure and Reliable Federated Learning using Blockchain. *arXiv* **2022**, arXiv:2201.11311.
12. Weng, J.; Weng, J.; Zhang, J.; Li, M.; Zhang, Y.; Luo, W. Deepchain: Auditable and privacy-preserving deep learning with blockchain-based incentive. *IEEE Trans. Dependable Secur. Comput.* **2019**, *18*, 2438–2455. [CrossRef]
13. Kuo, T.T.; Ohno-Machado, L. ModelChain: Decentralized Privacy-Preserving Healthcare Predictive Modeling Framework on Private Blockchain Networks. *arXiv* **2018**, arXiv:1802.01746.
14. Lu, Y.; Huang, X.; Zhang, K.; Maharjan, S.; Zhang, Y. Low-Latency Federated Learning and Blockchain for Edge Association in Digital Twin Empowered 6G Networks. *IEEE Trans. Ind. Inform.* **2021**, *17*, 5098–5107.
15. Zhang, Z.; Xu, K.; Li, Q.; Liu, X.; Li, L.; Wu, B.; Guo, Y. SecCL: Securing Collaborative Learning Systems via Trusted Bulletin Boards. *IEEE Commun. Mag.* **2020**, *58*, 47–53. [CrossRef]
16. Yang, Z.; Shi, Y.; Zhou, Y.; Wang, Z.; Yang, K. Trustworthy Federated Learning via Blockchain. *arXiv* **2022**, arXiv:2209.04418.
17. Benet, J. IPFS-Content Addressed, Versioned, P2P File System. *arXiv* **2014**, arXiv:1407.3561.
18. McMahan, H.B.; Moore, E.; Ramage, D.; Hampson, S.; Arcas, B.A.Y. Communication-Efficient Learning of Deep Networks from Decentralized Data. *arXiv* **2023**, arXiv:1602.05629.
19. Sun, T.; Li, D.; Wang, B. Decentralized Federated Averaging. *arXiv* **2021**, arXiv:2104.11375.
20. Hassan, F.u.; Ali, A.; Rahouti, M.; Latif, S.; Kanhere, S.; Singh, J.; AlaAl-Fuqaha; Janjua, U.; Mian, A.N.; Qadir, J.; et al. Blockchain And The Future of the Internet: A Comprehensive Review. *arXiv* **2020**, arXiv:1904.00733.
21. Bodkhe, U.; Tanwar, S.; Parekh, K.; Khanpara, P.; Tyagi, S.; Kumar, N.; Alazab, M. Blockchain for Industry 4.0: A Comprehensive Review. *IEEE Access* **2020**, *8*, 79764–79800. [CrossRef]
22. Witanto, E.N.; Lee, S.G. Cloud Storage Data Verification Using Signcryption Scheme. *Appl. Sci.* **2022**, *12*, 8602. [CrossRef]
23. Ma, C. Efficient Short Signcryption Scheme with Public Verifiability. In *Information Security and Cryptology*; Lecture Notes in Computer Science; Lipmaa, H., Yung, M., Lin, D., Eds.; Springer: Berlin/Heidelberg, Germany, 2006; pp. 118–129. [CrossRef]
24. Zhou, S.; Liao, M.; Qiao, B.; Yang, X. A Survey of Security Aggregation. In Proceedings of the 2022 24th International Conference on Advanced Communication Technology (ICACT), Pyeongchang, Republic of Korea, 13–16 February 2022; pp. 334–340. ISSN 1738-9445. [CrossRef]
25. Tu, X.; Zhu, K.; Luong, N.C.; Niyato, D.; Zhang, Y.; Li, J. Incentive Mechanisms for Federated Learning: From Economic and Game Theoretic Perspective. *arXiv* **2021**, arXiv:2111.11850.

26. Zou, J.; Ye, B.; Qu, L.; Wang, Y.; Orgun, M.A.; Li, L. A Proof-of-Trust Consensus Protocol for Enhancing Accountability in Crowdsourcing Services. *IEEE Trans. Serv. Comput.* **2019**, *12*, 429–445.
27. Oktian, Y.E.; Stanley, B.; Lee, S.G. Building Trusted Federated Learning on Blockchain. *Symmetry* **2022**, *14*, 1407.
28. Peng, Z.; Xu, J.; Chu, X.; Gao, S.; Yao, Y.; Gu, R.; Tang, Y. VFChain: Enabling Verifiable and Auditable Federated Learning via Blockchain Systems. *IEEE Trans. Netw. Sci. Eng.* **2022**, *9*, 173–186. [CrossRef]
29. Bayılmış, C.; Ebleme, M.A.; Çavuşoğlu, Ü.; Küçük, K.; Sevin, A. A survey on communication protocols and performance evaluations for Internet of Things. *Digit. Commun. Netw.* **2022**, *8*, 1094–1104. [CrossRef]
30. Pannekoek, M.; Spigler, G. Investigating Trade-offs in Utility, Fairness and Differential Privacy in Neural Networks. *arXiv* **2021**, arXiv:2102.05975.
31. Gould, S.J.J.; Cox, A.L.; Brumby, D.P. Diminished Control in Crowdsourcing: An Investigation of Crowdworker Multitasking Behavior. *ACM Trans. Comput.-Hum. Interact.* **2016**, *23*, 1–29. [CrossRef]
32. Valente, R.; Senna, C.; Rito, P.; Sargento, S. Embedded Federated Learning for VANET Environments. *Appl. Sci.* **2023**, *13*, 2329.
33. Rückel, T.; Sedlmeir, J.; Hofmann, P. Fairness, Integrity, and Privacy in a Scalable Blockchain-based Federated Learning System. *arXiv* **2021**, arXiv:2111.06290.
34. Liu, S.; Wang, X.; Hui, L.; Wu, W. Blockchain-Based Decentralized Federated Learning Method in Edge Computing Environment. *Appl. Sci.* **2023**, *13*, 1677.
35. Gas and Fees. Available online: https://ethereum.org/en/developers/docs/gas/ (accessed on 17 April 2023).