Reputation System based on Distributed Ledge to Secure DecentralizedFederated Learning

Jan von der Assen

vonderassen@ifi.uzh.ch

University of Zurich

Sandrin Raphael Hunkeler

University of Zurich

Alberto Huertas Celdran

University of Zurich

Enrique Tomas Martinez Beltran

University of Murcia

Gérôme Bovet

Federal Department of Defence, Civil Protection and Sports

Burkhard Stiller

University of Zurich

Research Article

Keywords: Decentralized Federated Learning, Reputation, Secure Aggregation, Distributed Ledger

Posted Date: October 4th, 2024

DOI: https://doi.org/10.21203/rs.3.rs-4997851/v1

Additional Declarations: No competing interests reported.

Eur. Phys. J. C manuscript No.
(will be inserted by the editor)

Reputation System based on Distributed Ledger to Secure Decentralized Federated Learning

Jan von der Assen[a,1], **Sandrin Raphael Hunkeler**[b,1], **Alberto Huertas Celdrán**[c,1], **Enrique Tomás Martínez Beltrán**[d,2], **Gérôme Bovet**[e,3], **Burkhard Stiller**[f,1]

[1]Communication Systems Group CSG, Department of Informatics IfI, University of Zurich UZH, 8050 Zürich, Switzerland
[2]Department of Information and Communications Engineering, University of Murcia, 30100 Murcia, Spain
[3]Cyber-Defence Campus within Armasuisse Science & Technology, 3602 Thun, Switzerland

Received: date / Accepted: date

Abstract Machine Learning (ML) faces several challenges, including susceptibility to data leakage and the overhead associated with data storage. Decentralized Federated Learning (DFL) offers a robust solution to these issues by eliminating the need for centralized data collection, thereby enhancing data privacy. In DFL, distributed nodes collaboratively train an ML model by sharing model parameters rather than sensitive data. However, DFL systems are vulnerable to poisoning attacks, where malicious participants manipulate their local models or training data to compromise the overall model. Existing robust aggregation methods attempt to mitigate these threats by evaluating the quality of models based on specific criteria before and during aggregation. However, these methods rely solely on the local perspectives of individual DFL participants, limiting their effectiveness in identifying malicious actors. More specifically, the role of Distributed Ledger technology in providing a reputation-based aggregation approach for decentralized learning has not been explored. Moreover, experiments with reputation-based attacks have not been performed. Thus, this work introduces a ledger-based reputation system that enables participants to share their local reputation assessments, which are then combined into a reputation score. This score informs a robust aggregation algorithm, facilitating weighted aggregation. Experimental results demonstrate that the proposed system effectively mitigates model poisoning attacks and defenses against attacks targeting the reputation system itself. Additionally, resource utilization metrics reveal trade-offs and scalability limitations, with the reputation system providing valuable information to participants while maintaining competitive latency levels.

Keywords Decentralized Federated Learning · Reputation · Secure Aggregation · Distributed Ledger

1 Introduction

The usage of Machine Learning (ML) in Artificial Intelligence (AI) has spurred innovations in several domains, leading to rapid adoption of AI. As such, AI has been deployed to solve complex problems in health care, insurance, retail, and the automotive industry [1]. The advent of off-premise, service-based access models, such as the ones offered by Large Language Models (LLM), has accelerated the adoption rate even further. For example, ChatGPT, one service relying on LLMs, has gained 100 million active monthly users in a record-breaking time of two months after its release to the public [2].

Traditional ML techniques, including novel forms such as LLMs, present a common characteristic: data is needed both for training and inference. Thus, these systems are built and operated on the premise that users agree on having their data used and transmitted to a central location. Hence, data privacy is not ensured at development and runtime, making these approaches unsuitable for privacy-preserving scenarios. Federated Learning was proposed in 2016 as an approach to deal with this situation and comply with data privacy regulations that forbid the collection of certain data in the first place. In federated learning, training data is not held in a central location. Instead, a distributed system is formed by several nodes operating on local data to achieve a common goal: training a global model that comprises the characteristics of the training data without actually revealing that data. This can be achieved in two variants: sharing the model

[a]e-mail: vonderassen@ifi.uzh.ch (corresponding author)
[b]e-mail: sandrinraphael.hunkeler@uzh.ch
[c]e-mail: huertas@ifi.uzh.ch
[d]e-mail: enriquetomas@um.es
[e]e-mail: gerome.bovet@armasuisse.ch
[f]e-mail: stiller@ifi.uzh.ch

parameters with a central orchestrator or sharing them with a set of nodes in a decentralized manner [3,4].

Decentralized federated learning (DFL) successfully eliminates the necessity of a central coordinator constituting a single point of failure [5]. In the past, many decentralized systems have emerged for various areas, such as finance and energy. The proliferation of these systems has led to analyses and discussions of them as not just technical machines but rather as socio-technical machines. Applying the same reasoning to DFL raises several concerns, including security risks such as model poisoning or inference attacks [6].

The literature has proposed several approaches to secure DFL against these attacks [7]. In this sense, robust aggregation, where the exchanged models are analyzed based on certain criteria, is a common approach. However, when combining reputation-based approaches that model the opinions of the participants in the network, a more complex view of the participants and their behavior is needed. For example, even though a node may not interact with a malicious one, it could still form an opinion about its behavior from other nodes that have been assessed as trustworthy based on previous model exchanges. Through this, a node could learn about the trustworthiness of nodes without analyzing their models. Depending on the federation topology, this could influence the performance of the system as a whole since a full mesh topology – where everyone could analyze every other node's models – may unlikely scale for larger DFL networks.

In such a reputation system, new threats arise since the reputation system constitutes an increase in the attack surface. For example, malicious nodes can execute Denial of Service (DoS) or reputation-based attacks. Hence, this paper analyzes how distributed ledger (DL) technology can be integrated into a DFL network to enable nodes to securely share opinions, gather a global view of each node (*i.e.,* a trust representation), and then adapt the model aggregation process to accommodate for trust. In the current state-of-the-art, two key limitations exist. Firstly, no studies have explored how DLs can be applied when employing Federated Learning in a decentralized setting. Secondly, reputation-based FL approaches have not investigated the effectiveness of their systems against reputation-based attacks. Thus, this article presents the following contributions:

- A reputation-based framework that comprises a DFL network with several aggregators and a DL network involving an Oracle to deploy smart contracts and the reputation system smart contract.
- The prototypical implementation of the framework's key components and their integration into FedStellar, a platform for research on DFL. In this context, a Flask web app implements cross-cutting concerns of the DL Oracle. Furthermore, the reputation system is implemented in Solidity, comprising an adjacency matrix to store opinions and an EVM-compliant implementation to compute the overall reputation values.
- A set of experiments considering the framework in a scenario with several DFL nodes collaboratively training a multi layer perception (MLP) model using the MNIST dataset with Non-IID distribution. More in detail, the following four experiments were executed: *(*i) an analysis of the relative and absolute computational resources required by the DFL nodes, DL validators, non-validators, front-end, boot nodes, and Oracles; *ii)* an evaluation of the aggregation time delays and financial costs incurred by adopting DL technology within the DFL scenario; *iii)* the execution of several attacks to establish the defense effectiveness of the reputation system against model poisoning attacks; and *iv)* the execution of badmouthing attacks poisoning the reputation system itself to assess the defense robustness of the reputation algorithm.

The remainder of this article is structured as follows. Section 2 introduces key concepts and reviews related studies. Section 3 presents the framework architecture and prototypical implementation. The efficiency and effectiveness of the approach are evaluated in Section 4; leading up to concluding considerations in Section 5.

2 Background and Related Work

This section starts introducing the main aspects of Federated Learning and Distributed Ledgers to later review the literature done combining these fields.

2.1 Federated Learning

To collaboratively refine the Google Keyboard while maintaining user privacy, Google proposed FL in 2016. At the core, FL involves that instead of sharing raw or pre-processed training data, only the model weights are exchanged. Several roles are typically observed in a federation: Clients use the resulting models in the federation and, optionally, use them as a base for retraining, using locally accessible training data. Aggregators receive the updated models from one or more clients and aggregate them into a new global model. In turn, they select clients to broadcast the new model [8]. When analyzing the roles and responsibilities of the nodes in a given federation, the overall scenario can be classified: centralized federated learning involves a single, often predetermined aggregator at a central position. DFL, on the other hand, does not exhibit a pre-determined or centralized entity, paving the way for different topologies [3]. In addition, a federation can be characterized by its training data distribution (*i.e.,* yielding vertical or horizontal learning approaches) [5].

Various application areas are said to benefit from DFL such as industrial engineering, health care, or mobile computing. Similarly, the research community has described several threats during the training, intercommunication, and inference. Due to the lack of a centralized controller, any entity can perform model poisoning attacks, where the model's parameters are adapted during training using different methods. As in any FL architecture, DFL is susceptible to data poisoning attacks since data is held at the client's side. Thus, different strategies, such as label flipping or backdoor attacks, exist. Although privacy is a key concern in FL, inference attacks target the confidentiality of the data that was used to train. More specifically, the literature describes strategies such as membership inference and reconstruction attacks [7].

Although all attacks are relevant in DFL, this work focuses on attacks targeting the integrity, correctness, and availability of the federation's model. As such, inference attacks are out of the scope of the work at hand. Both IID and non-IID scenarios are considered to defend against the attacks. It is expected that a minority subset (*i.e.,* less than 50%) of nodes participating in the federation are malicious and the remaining ones are honest. Furthermore, a realistic threat model of a secure DFL scenario must address threats relating to any additional defense mechanism – in this scenario, the reputation system. For example, Denial-of-service (DoS) attacks may temporarily hinder the availability of the reputation system. Badmouthing attacks aim to decrease the reputation of honest nodes or increase the reputation of malicious ones. Finally, attacks on the underlying communication system are not the focus of the current research.

2.2 Distributed Ledgers

Distributed ledger technology (DLT) is a decentralized application that enables secure, transparent, and trustworthy transactions in an environment where honesty can't be assumed [9]. The concept of DLT was first introduced in 2008 by Satoshi Nakamoto through the Bitcoin white paper, aiming to resolve the Byzantine generals' problem [10]. This issue describes the challenge faced by multiple honest leaders attempting to reach a consensus while being disrupted by dishonest ones [11].

To address this problem, DLT relies on a consensus protocol rather than a trusted third party to establish trust among participants. Consensus protocols govern how nodes validate transactions, ensuring that all nodes agree on the same information and maintain a synchronized ledger [12, 9, 13]. DLT operates within a peer-to-peer (P2P) network, forming an append-only database (ledger) that is simultaneously maintained and stored by distributed nodes. This design removes single points of failure, making the ledger resilient to node breakdowns [9, 12].

Within DL technology, a blockchain is considered a decentralized ledger where all transactions are recorded permanently and cannot be altered [14]. The first decentralized ledger, Bitcoin (BTC), was built on top of a blockchain [10]. This distributed ledger consists of batches of valid transactions, known as blocks, which contain metadata such as the Merkle tree root, the hash of the prior block, and consensus protocol parameters [15]. The linked block hashes give the blockchain its name and serve as a unique identifier for each block while ensuring their integrity [12]. If a transaction were modified, the hash of its corresponding block, as well as those of all subsequent blocks, would change [16]. Combining this with the highly duplicated and distributed ledger makes blockchain technology highly resistant to tampering and thus perceived as immutable.

One technology enabled by DLs is smart contracts, which are self-executing programs that run on a blockchain, ensuring their correct execution through the underlying consensus protocol [17, 12]. These contracts are duplicated and stored across all participating blockchain nodes [18], making them nearly immutable against tampering. Ethereum was a pioneer in introducing the development and deployment of smart contracts [19]. Each interaction with a smart contract that modifies the ledger is recorded as a transaction and stored permanently on the blockchain [14]. Smart contracts are used to build distributed applications (dApps), which host parts of their back-end and database on a blockchain [13]. Ideally, dApps should not rely on human interaction and have all policies encoded in their smart contracts [16]. Centralized applications use login credentials for authorization at a server, whereas dApps utilize wallet addresses and private keys of blockchains for authentication. While dApps provide more transparency and improved identity verification, they also introduce challenges such as execution efficiency and the irreversibility of exploited vulnerabilities [18].

2.3 Literature Review

Based on the literature review, eleven relevant studies have been analyzed, as outlined in Table 1. Four aspects have been elicited to contrast the approaches. First, the usage of the distributed ledger within the Federated Learning context. Secondly, the security function achieved by the integration is analyzed (*e.g.,* whether the DL is used to filter a model or a node). Thirdly, it was assessed whether approaches focus on DFL, or on its centralized counterpart. Finally, it was sought whether a prototypical implementation of the approach exists.

With respect to DL usage, the examined frameworks utilized different tools within the distributed ledger to leverage its trusted, secure, and tamper-resistant characteristics. Integrating CFL aggregation into a smart contract or embedding the process within the consensus mechanism [20, 21, 27, 28,

Table 1: Literature Review of Distributed Ledger Technology in Federated Learning

Work	*DL Usage*	*Security*	*Architecture*	*Implementation*
[20] 2023	Consensus	Con-dBFT	Decoupled, CFL	HLF
[21] 2021	Consensus	PoW	Coupled, CFL	Public Ledger
[22] 2020	Reputation	Node Filtering	Decoupled, CFL	HLF
[23] 2019	Reputation	Node Filtering	Decoupled, CFL	BC, IPFS
[24] 2024	Aggregation	Partitioned Model	Decoupled, CFL	HLF
[25] 2021	Analytics	Anomaly Detection	Decoupled, CFL	Private BC
[26] 2023	Incentivization	Model Filtering	Decoupled, CFL	HLF
[27] 2023	Consensus	dBFT	Decoupled, CFL	Exonum
[28] 2019	Aggregation	PoW or pBFT	Decoupled, CFL	HLF
[29] 2023	Reputation Aggregation	Node and Model filtering	Decoupled, CFL	Simulation
[30] 2022	Aggregation	Multi-Layer BC	Semi-decoupled, DFL	Flask and ETH
This article	Reputation	Weighted Aggregation	Decoupled, DFL	Private PoA ETH, FedStellar

ETH=Ethereum, HLF=Hyperledger Fabric, BFT=Byzantine Fault Tolerance,
IPFS=InterPlanetary File System, PoW=Proof of Work, PoA=Proof of Authority

30] closely links the architectures of FL and blockchain, causing changes in one to affect the other directly. Conversely, more loosely connected methods, like incentive systems [26], reputation systems [22,23,29], and analytic systems [25], are less reliant on the FL process. Here, the actual training is decoupled from the DL, which can provide optional benefits, such as increased participation or improved insight into the network. Thus, the DL also does not pose as a critical dependency for the overall system.

The *Security* column outlines each framework's primary technological or logical approach to minimizing its attack surface. The reviewed frameworks generally employ four key strategies to mitigate threats such as poisoning attacks or reduce the likelihood of successful attacks overall: consensus protocols, filtering, anomaly detection, and weighted aggregation. Integrating aggregation into consensus algorithms [20,21,28,27] hinges on nodes' ability to verify the validity of transactions. However, this method still relies on computing a trust metric to assess the quality of contributions. Additionally, the effectiveness and applicability of Proof-of-Work-based protocols (PoW) in mitigating attacks raises questions, particularly in small FL networks and on low-power devices. Opposedly, information from the DL could be used for network censorship: filtering out models or participants [22,23,26,29], has been shown to improve model quality and training performance. However, this approach assumes uniform data distribution and is vulnerable to filtering out false positives. Both strict filtering and weighted aggregation are highly dependent on evaluating the quality and performance of individual models. These methods could be enhanced by incorporating reputation metrics to increase the robustness and fairness of the aggregation algorithms.

With respect to the architecture followed by the approaches, varying degrees of coupling between worker- and blockchain nodes are observed. Coupled nodes are streamlined but challenging to extend, while decoupled nodes are easier to modify but more resource-intensive. Semi-coupled architectures combine these approaches. Coupled nodes are suitable for productive environments prioritizing persistence and resources, while decoupled nodes are better suited for development and research settings where experimentation and adaptability are key. Aside from [30] and [21] studies follow a decoupled architecture. Moreover, a clear lack of approaches focusing on DFL is apparent. Aside from [30], no approaches investigate the suitability of DL technology outside of CFL.

The implementation of these approaches refers to their underlying technological basis. Two popular open-source distributed ledger frameworks, Ethereum and Hyperledger Fabric (HLF), were observed. HLF offers full configurability but requires extensive knowledge of encryption methods and blockchain protocols, making it suitable for research frameworks focusing on consensus protocols. In contrast, Ethereum's open-source implementations are easier to use and deploy but may lack extensibility and modularity.

In summary, previous research has focused on decentralizing central aggregating entities in CFL using DL technology and employing trust/reputation metrics. However, these studies did not explore the effectiveness of computing and storing reputation values in DFL. Additionally, existing frameworks primarily addressed poisoning attacks excluding attacks on the reputation system itself. Thus, this work aims to fill these gaps by developing a ledger-based reputation system for DFL, implementing a reputation-based weighted aggregation algorithm, and evaluating the robustness of the reputation system against adversarial attacks and reputation attacks.

3 Architecture

This section presents the overall architecture of the proposed reputation system and its integration into FedStellar [31], an

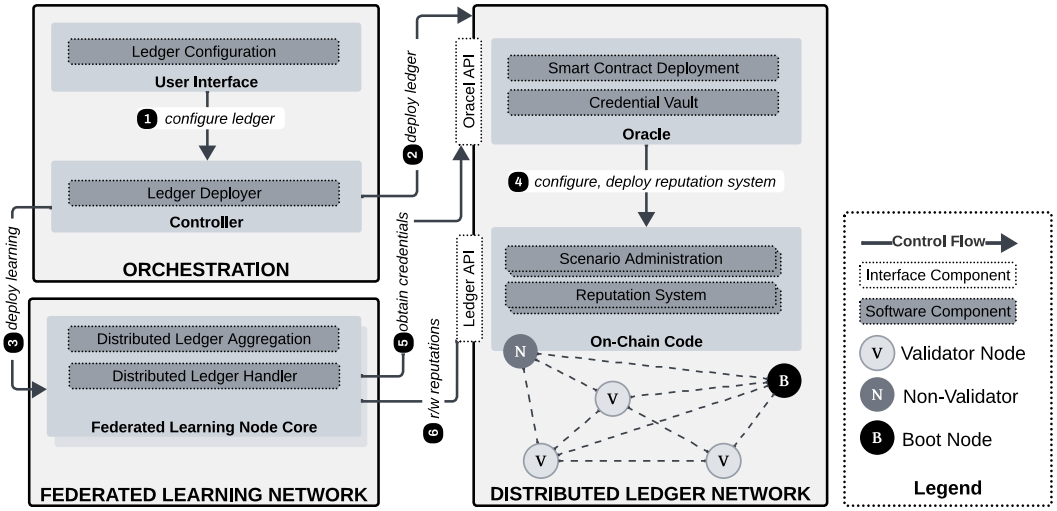


Fig. 1: High-level View on the Distributed Ledger-based Architecture

existing DFL platform. Specifically, the design and implementation of two key reputation components are introduced: the DL-based aggregation algorithm and the reputation system, which is its on-chain counterpart.

As highlighted in Fig. 1, the components of the proposed framework are introduced in six steps. Since Fedstellar already provides a front- and back-end to manage, configure, and execute DFL scenarios, its components are extended to provide the desired functionality. First, in the *User Interface*, the dashboard is extended to configure the DL and its reputation system. For example, it enables the configuration of the aggregation algorithm. The User Interface sends all configuration aspects to the Controller (step 1 in Fig. 1). Here, an additional component is needed to act as a *Ledger Deployer*. This component must execute two tasks: provisioning the DL infrastructure (step 2) and deploying the necessary infrastructure for the DFL scenarios (step 3).

3.1 Distributed Ledger Network

The proposed DL infrastructure consists of four components running on a generic DL network. As it is common in DL networks, they may require several different nodes to operate. Herein, three DL network nodes are assumed. Boot nodes do not actively engage in the DL's consensus mechanism – they enable node discovery in a Peer-to-peer (P2P) setting. Thus, the validator and non-validator nodes can discover and interact with each other. Validator nodes provide key aspects, such as immutable data storage and the operation of the consensus algorithm, which ultimately enables write operations through transactions. Only the non-validator nodes present an API for clients that do not implement a DL node. Nevertheless, they still fully engage in the synchronization of state within the DL. Thus, they act as a gateway to the DL. To present a single access point deploying the reputation system, an *Oracle* is proposed. Indeed, for a public, permission-less Blockchain, the Oracle would represent a centralized component. Thus, it must be assumed that there is a centralized entity that is considered trustworthy, at least for the deployment of the system – after provisioning, it does not actively engage in the operation. The Oracle compiles and deploys the reputation system chaincode and provides funding for the DFL nodes (step 4 in Fig. 1). In a public permission-less scenario, nodes could provide their own funding and cryptographic material. Thus, any credential management applies only to experimental settings.

Algorithm 1: get_reputations(list<node> names)

```
filter_out_unknown_nodes(names) reputations ←
 list<reputation> foreach target ∈ names do
    sum_reputation ← 0 n_reputation ← 0 foreach node ∈
     registered_nodes do
        if confirmed_neighbors(node, target) then
            sum_reputation ← sum_reputation +
              avg_opinion(of=node, about=target)
            n_reputation ← n_reputation + 1
        end
    end
    reputations.push(node=target, value=sum_reputation /
     n_reputation);
end
median ← median(reputations) stddev ← stddev(reputations)
 if stddev > constant_a then
    foreach reputation ∈ reputations do
        n_stddev ← abs(median - reputation.value) / stddev
         if n_stddev > const_b then
            reputation.value ← reputation.value / (n_stddev
              * constant_c)
        end
    end
end
return reputations;
```

The on-chain code notarizes the configuration of the federation and provides the reputation system. This is achieved by implementing a decentralized application to record *opinions* and synthesizing a global *reputation* of those values. The first functionality is achieved by allowing each node to store opinions in an adjacency matrix. More complex is the formation of a reputation from those opinions: In Algorithm 1, the key steps of the reputation system are represented. Nodes may receive models from a sub-set of directly connected nodes (*i.e.,* neighbors). The algorithm receives the addresses of those nodes as only input – for each of those nodes (referred to as target node), the reputation is calculated as follows. First, unregistered node names are removed. Then, for those nodes from whom a model exchange with the target node actually occurred (*i.e.,* the `confirmed_nodes`), the average pushed opinion of the target node is computed. This average represents an initial reputation value, which is now established for each of the nodes. Hence, a global view of the reputations of the nodes exists. Based on these average opinions, the median and standard deviation of the global reputation distribution are obtained. Since high deviations in opinions might indicate an active poisoning attack, such nodes are actively punished: if the number of standard deviations it differs from the median reputation is above a certain threshold, the final reputation value is scaled by dividing it by the number of standard deviations multiplied by a constant penalty factor.

3.2 Federated Learning Network

These DL-based components (*e.g.,* the blockchain network and the on-chain reputation system) form a distributed backend for secure DFL. Thus, the DFL components must be able to interact with the DL backend. To do so, the DFL components can be bootstrapped through the Oracle, enabling them to obtain the credentials to participate in the reputation system (step 5 in Fig. 1). During the training phase, the DFL nodes interact with the DL through the *Distributed Ledger Handler*. Essentially, this algorithm acts as a drop-in replacement for other secure aggregation mechanisms implemented in [31], such as *FedAvg* or *Krum*. Algorithm 2 details how the model aggregation is weighted according to the reputation system's state using the set of received models as input. First, for each sender, the model is evaluated using a trustworthiness opinion metric. As a reference, cosine similarity is used to compare models, assuming poisoned models exhibit high dissimilarity, as shown in Equation 1.

$$\cos\_\mathrm{sim}(A,B) = \frac{a_1 \cdot b_1 + a_2 \cdot b_2}{\sqrt{a_1^2 + a_2^2} \times \sqrt{b_1^2 + b_2^2}} \tag{1}$$

Based on this metric, a convex transformation is applied by exponentiating with a constant value – effectively magnifying differences for higher dissimilarity while smoothing small differences. These values are then written to the on-chain reputation system by accessing the Smart Contract through the API provided by the non-validator nodes (step 6 in Fig. 1). Opposed to the reading of the opinions, this represents a write operation to a distributed system, which can lead to delays since the system must achieve a consensus state. In the experiments, the effect of introducing this operation is assessed. Subsequently, the new global reputation values of the current training phase must be obtained. The remaining steps of the algorithm then apply a weighted aggregation, contrasting other approaches that use such a value to filter nodes or models.

Algorithm 2: aggregate(list<model> models)

```
local_model ← models[self];
metrics ← {};
foreach model ∈ models do
    if model ≠ local_model then
        metrics[model.sender] ←
          cosine_similarity(local_model, model);
    end
end
local_opinion ← {};
foreach (sender, similarity) ∈ metrics do
    local_opinion[sender] ← transform(similarity);
end
blockchain_handler.push_opinion(local_opinion);
senders ← {};
foreach model ∈ models do
    senders ← senders ∪ {model.sender};
end
reputations ← blockchain_handler.get_reputations(senders);
sum_reputations ← 0;
foreach reputation ∈ reputations do
    sum_reputations ← sum_reputations + reputation;
end
normalized_reputations ← {};
foreach reputation ∈ reputations do
    normalized_reputations[reputation.name] ← reputation /
      sum_reputations;
end
final_model ← zero_copy(local_model);
foreach layer ∈ final_model do
    foreach model ∈ models do
        final_model[layer] ← final_model[layer] +
          model[layer] *
          normalized_reputations[model.sender];
    end
end
return final_model;
```

3.3 Prototypical Implementation

To implement a prototype of the previously described architecture, the Fedstellar platform [31] was employed, inheriting several implementation decisions. As such, most com-

ponents followed the respective Python-based implementations and related libraries. The *Ledger Deployer* was implemented as a Python-based RESTful API. Similarly, the User Interface was extended, integrating the necessary configuration steps into the Flask-based frontend [31]. To integrate the DL infrastructure, the go-ethereum (geth) client of the Ethereum project was leveraged [32]. For the initial experimentation, a setup was implemented using the Proof-of-Authority consensus mechanism. Here, it was needed to implement the DL infrastructure and the controller to easily deploy the nodes, as well as fund addresses, compile and deploy the smart contract. These aspects are made available through a modular prototype in [33], while the overall architecture is available in [31].

Two implementation details are essential; the threshold to consider a node as potentially malicious was implemented in a dynamic manner, as shown in Listing 1. Furthermore, in the *Distributed Ledger Aggregation*, an exponent of $n = 3$ was used to apply the convex transformation.

```
if (stddev >= 5 * MULTIPLIER &&
    stddev_count >= 1 * MULTIPLIER &&
    reputations[i].reputation > 0) {

      uint64 divisor = (2 * stddev_count)**2
                    / MULTIPLIER;
       reputations[i].reputation =
       ((reputations[i].reputation
       * MULTIPLIER) / divisor);
}
```

Listing 1: Reduction of Highly Deviating Reputation Values

4 Evaluations

This section first evaluates the Blockchain Network's performance and resource utilization. Then, it measures the Reputation System's ability to detect and mitigate poisoning attacks.

All experiments measuring resource utilization have been conducted under the same hardware and settings of FedStellar. For running FedStellar, a virtual machine with 62 GiB of memory and an AMD-EPYC 32-core processor was assigned. The underlying scenario consists of ten participating DFL nodes arranged in a full mesh topology, and they train a multi-layer perceptron model (MLP) on the MNIST dataset with IID and Non-IID distributions (the later implemented using Dirichlet with an alpha value set to $\alpha = 0.5$). The training considers ten rounds with one epoch each and a batch size of 32. The Blockchain Network consists of three Validator Nodes with a block time of one second. Using less than three Validator Nodes drastically reduces the stability of the PoA consensus algorithm, which was the reason for omitting those configurations. Further, the Blockchain Network contains an Oracle, a Non-Validator Node, and a Boot Node. Other than the number of participants or nodes, the chosen settings reflect FedStellar's current default parameters.

4.1 Attacks

This section evaluates the Reputation System's effectiveness while detecting and mitigating the impact of model poisoning and badmouthing attacks. To achieve this, the scenario involving ten federated nodes, as described at the beginning of the section, was used.

4.1.1 Model Poisoning

A model poisoning attack on FedStellar was executed using the existing Noise Injection Attack in FedStellar. This implementation designates malicious nodes in the federation, which are responsible for aggregating all received models using a benign aggregation algorithm before poisoning the newly aggregated model. After poisoning the model locally, the malicious nodes continue training the model honestly. The model poisoning attack is carried out by adding Gaussian-distributed noise. Specifically, a random Gaussian distribution with the same dimensions as the model is generated and scaled by a static factor before being summed with the model to introduce the poisoning. The Gaussian distribution is generated with a mean of zero and a variance of one. Then, this normal distribution is scaled up by a factor of ten, preserving the mean but increasing the variance to 100.

For the Non-IID scenario (Dirichlet with an alpha value set to $\alpha = 0.5$), Fig. 2 shows the change in accuracy with the individual aggregation algorithms by increased number of malicious nodes. As can be seen, the Blockchain Reputation (DL-DFL) accuracy remained stable and high with increased percentage of malicious nodes. FedStellar's implementation of the Krum aggregation algorithm shows a sim-

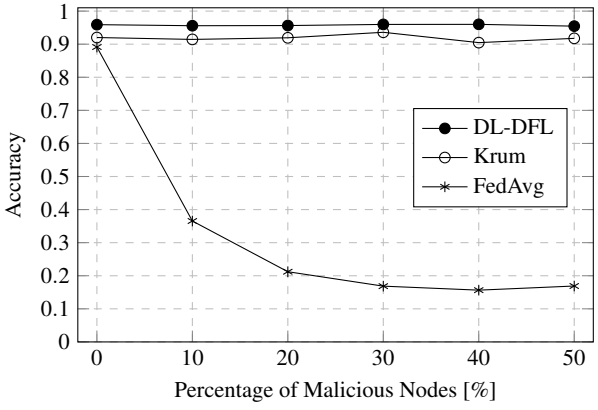


Fig. 2: Performance Comparison during Noise Injection (Non-IID)

ilar resilience against model poisoning but is outperformed by the Blockchain Aggregator. In addition, the FedAvg aggregation algorithm does not offer a robust poisoning defense. Therefore, its accuracy rapidly declines with nodes injecting noise. Furthermore, the accuracy of the final model with the Blockchain Aggregator outperformed both Krum and FedAvg without active poisoning.

In order to understand better how DL-DFL works, Fig. 4 (b) shows a heatmap of the average reported opinion by node about each other node in the previously used Non-IID scenario with four model poisoning nodes over ten rounds. The axes represent the malicious and benign nodes, which are arranged in alphabetical order according to their identification index. Since the local opinion is computed using similarity metrics, the values are also an indicator of similarities of the node's models. The heatmap shows two rather consistent groups reporting high opinion values for nodes of their group while reporting low opinion values about the other group's nodes. This indicates that models actively poisoned by malicious nodes are more similar to each other than to the benign nodes' models and vice versa. Surprisingly, the reported opinions of the malicious nodes about each other are considerably higher than the opinions of the benign nodes about each other. More in detail, two nodes, *Benign 1* and *Malicious 4*, deviate from their individual group. Both are evaluated as semi-honest by their individual group as well as by the other group. This shows a partial success of node *Malicious 4* in maintaining a constant rate of poisoning while being partially accepted by the benign nodes for aggregation. At the same time, node *Benign 1* was successfully poisoned by deviating from its group and sharing similarities with actively poisoned nodes' models.

The proposed solution computes a subjective reputation score for each individual node. Therefore, in scenarios with active poisoning, it creates encapsulated groups which highly agree on their quality of contribution. The groups mainly aggregate with other nodes of their group while decreasing the aggregation weights for nodes of the other group. This reduces the necessity of classifying the individual groups as benign or malicious in the Reputation System.

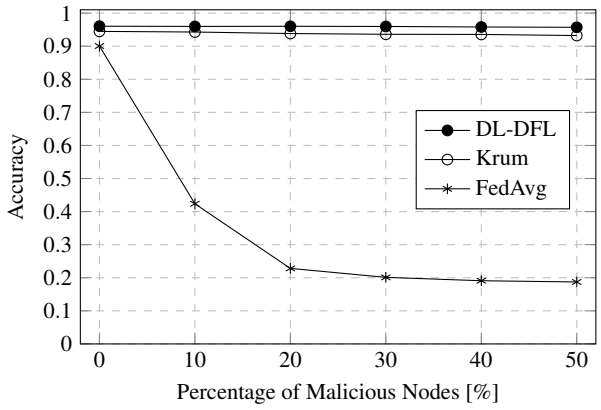


Fig. 3: Performance Comparison during Noise Injection (IID)

Regarding the IID scenario, Fig. 3 shows the change in accuracy with the individual aggregation algorithms by an increased number of malicious participants or nodes. In this case, the Blockchain Reputation (DL-DFL) accuracy remained stable and high with an increased percentage of malicious participants. Similarly to the Non-IID scenario in Fig. 2, the Krum aggregation algorithm achieves a similarly high accuracy. Overall, the accuracy of the final model with DL-DFL outperformed both Krum and FedAvg without active poisoning in both Non-IID and IID scenarios.

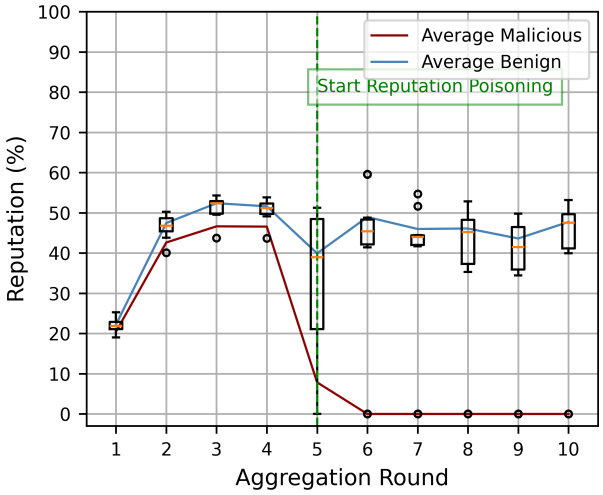
4.1.2 Reputation Attacks

Despite the advantages reported in the previous experiment, it is important to mention that the Reputation System introduces new vulnerabilities by depending on the honest participation of the nodes. High deviations in the reputation values reduce the Reputation System's ability to detect and mitigate model or data poisoning attacks. Consequently, multiple malicious attackers flooding the Reputation System with randomly generated opinion values could make the Reputation System unusable. To evaluate this aspect, ten participating nodes were set to train a multi-layer perceptron model (MLP) on the MNIST dataset with a non-IID distribution. As in the previous experiment, the training consisted of ten rounds with one epoch each and a batch size of 32.

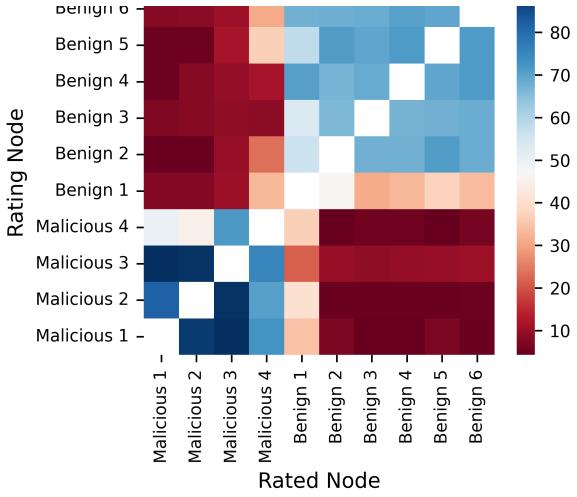
Fig. 4 shows the average computed reputation for both the reputation poisoning nodes in red and the benign nodes in blue. The box plots visualize the distribution of the computed reputation for all nodes in each round. Rounds not affected by reputation poisoning show a stable and uniform average reputation for all nodes. As of round five, the randomly generated opinion values start disturbing the reputation of all participating nodes. It means that the Reputation System was able to automatically recognize the anomaly, reducing the reputation of the malicious nodes to zero. This results in the malicious node's models being excluded from all further aggregations performed by honest nodes.

4.2 Distributed Ledger Overhead

The previous experiments established the effectiveness of the system as a defense system. However, since this is achieved by integrating elements of DL technology, the newly introduced overhead must be assessed – this section presents the resource and time delay overhead of the overall system. For these experiments, the same setup has been used.



(a) Global Reputation



(b) Reputation Scores

Fig. 4: Reputation Attack Effects

4.3 Computational Resources

Fig. 5 highlights the amount of CPU and memory used by the different types of nodes to provide infrastructure, model training, and validation. As indicated, the large majority (between 99.8% and 97.2% for CPU time and 95.0% and 85.5% for memory) of resources are spent on model training (*i.e.,* on the DFL *Cores*). As such, the DL-related aspects may not introduce a bottleneck, especially since such a DFL scenario must likely already consider computationally capable devices to engage in model training. Excluding the resource-intensive model training components reveals that the validator nodes require the largest amount of CPU and memory. In absolute numbers, 31.68 MiB of memory and 9.19 seconds of CPU time are consumed for the set-up described at the beginning of this section.

4.4 Aggregation Delay

Since the DL does not represent a resource bottleneck, the effect of the system on the aggregation time is analyzed since the aggregation algorithm comprises synchronous read and write operations to the reputation system. Two operations are needed to interact with the reputation system: new opinions are written to the contract, while the new global reputation values are requested to weigh the model updates. The finality of block synchronization in the DL network introduces varying degrees of delay. For example, if an opinion value is written shortly before a new block is created (*i.e.,* at the upper bound of the block time), the incurred delay is low. Thus, the block time (*i.e.,* the configuration of the DL with respect to its synchronization interval) influences the aggregation time, leading to a positive correlation.

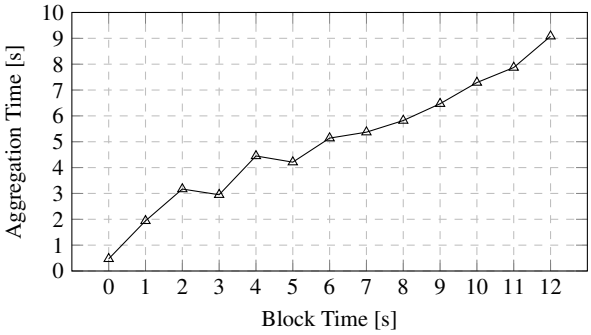
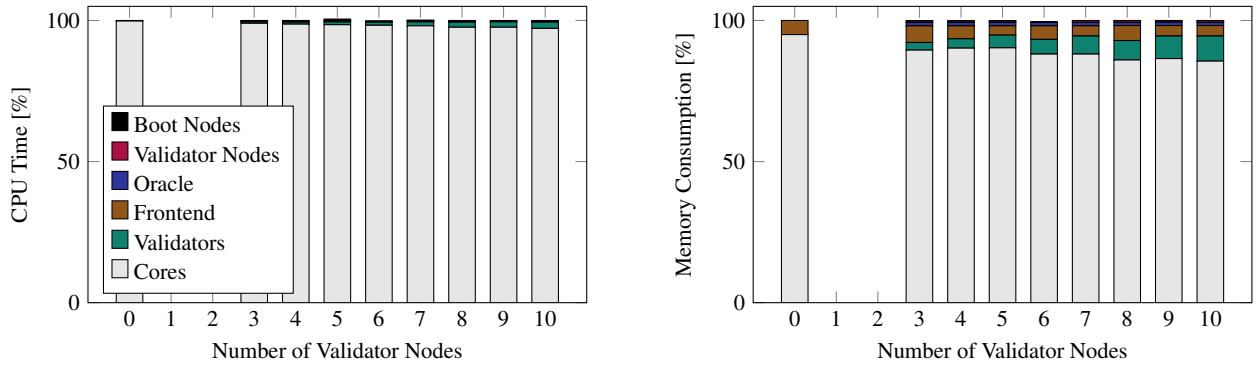


Fig. 6: Influence of Block Time to Aggregation Time

Fig. 6 plots the relation between block and aggregation time. If the block time is set to zero, any transaction sent to the DL is immediately validated and a new consensus must be achieved based on it. As such, it represents the lower bound at which a DL might act as a reputation system. However, such a low block time would likely introduce vulnerabilities in the DL infrastructure. Nevertheless, it sheds on a potential lower bound of 0.47 seconds. Furthermore, these data enable a comparison with public, permissionless DLs (*i.e.,* Blockchains). For example, the block time of 12 seconds, which represents the upper bound that was established, resembles the one used in the Ethereum main network (excluding network delays). As established by the values obtained from these averages of ten experiments, a block time of 12 seconds led to an aggregation time of 9.08 seconds. As will be compared hereinafter, this presents a large overhead to existing aggregation algorithms. Thus, the tolerance of such a delay - and thereby the applicability of a public, permissionless DL - must be evaluated for a particular DFL scenario. For example, a scenario with long-running iterations may be able to accommodate this delay. Table 2 compares two prominent aggregation algorithms and the DL-



(a) Relative CPU Consumption

(b) Relative Memory Consumption

Fig. 5: Relative Resource Consumption for Various DL Configurations

based one against *FedAvg*. All algorithms that aim to improve the resilience of the DFL network lead to a stark increase in aggregation time. However, even when the block time is set to its lower bound (*i.e.,* zero seconds), the resulting aggregation time of 0.47 seconds represents a 42% increase over *Krum*.

4.5 Gas Cost

Since the DL infrastructure must be operated without the direct involvement of a trusted third party, the computation within this network must be remunerated. Using the technologies involved in the system's development, the complexity of executions is measured in *gas*. Thus, in addition to the computational resources and time delays, this financial aspect must be established. To provide a more meaningful discussion, the gas costs of the Ethereum mainchain are leveraged. However, it must be emphasized again that other DL networks (*e.g.,* private permissioned ones) could serve as an alternative. In that sense, the cost approximations may present an upper-bound estimate. To quantify the gas costs, the conversion price of Ethereum to USD and the gas cost of 27.3 Gwei per gas were collected on April 8, 2024.

As shown in Fig. 7, the gas cost increases with the number of nodes participating in the federation. As defined in Section 3, opinions are stored in a matrix. In that sense, every node can publish an opinion about any other valid node in the federation. Thus, a quadratic relationship arises since the number of possible opinions (*i.e.,* node pairs) increases with each new node. In addition, each scenario comprises static cost to setup the federated and deploy the reputation system. As demonstrated in Fig. 7, the cheapest setup, comprising merely three training nodes, led to costs of 13 USD to execute one scenario. When increasing the number of nodes to ten, the cost increases to 79 USD. As such, organizations aiming to integrate DL technology to secure their DFL network must be aware of both factors: the absolute cost of maintaining one node and the increasing nature of the cost with a growing network. The applicability of DL technology in DFL must thereby be assessed for the particular scenario at hand.

Aggregation Time	***FedAvg***	***TrimMedian***	***Krum***	***DL+DFL***
Absolute	0.03s	0.10s	0.33s	0.47s
Relative	+0%	+233%	+1000%	+1466%

Table 2: Comparing Aggregation Algorithms with FedAvg

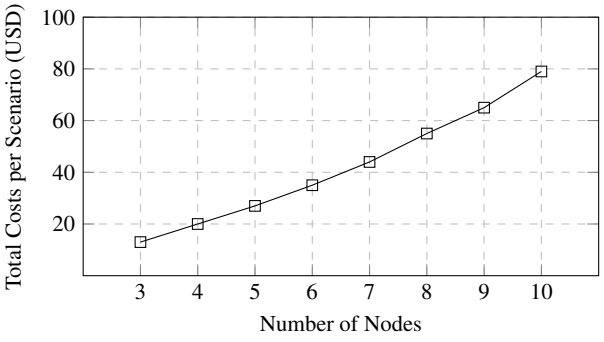


Fig. 7: Effect of Federation Size on Gas Cost

5 Conclusions and Future Work

This article introduced a framework for computing the reputation of trainers participating in DFL scenarios by using DL technology. The framework comprises three modules: *(1)* a blockchain scenario controller, *(2)* a DL network comprising an oracle and a reputation system, and *(3)* a reputation-based aggregation mechanism that uses the global reputation values to weigh model updates. To evaluate the efficacy of DL in providing a backbone for reputation-based DFL, the

framework was implemented into the FedStellar DFL platform.

Based on the fully functional prototype implementation, several experiments were performed. First, it was demonstrated that the DL-based system outperforms other aggregation approaches such as Krum or FedAvg. Furthermore, reputation attacks were executed, highlighting that the platform is robust even in the presence of such attacks. In contrast, the effectiveness of the platform was analyzed. This revealed that although the platform can indeed present more resilient learning, a considerable overhead in terms of aggregation time is introduced. Furthermore, a cost analysis revealed that the applicability of a system must be closely analyzed with respect to the socio-economic factors where it would be deployed. Here, both security aspects and financial requirements would need to be considered to assess whether it can be deployed in public, permission-less blockchains or in private, permissioned DL networks. Finally, it was found that the overhead in terms of resource consumption may be negligible compared to the model training activities.

In the future, further research directions exist: cost optimizations can investigate whether interaction with the reputation system could be executed in an asynchronous manner. While this would compromise accuracy in the short term, it might yield considerable performance improvements. Furthermore, the applicability of the platform for larger networks and different DL deployments will be analyzed.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

Ethical approval This article does not contain any studies with human participants or animals.

Acknowledgements This work has been partially supported by *(a)* the Swiss Federal Office for Defense Procurement (armasuisse) with the CyberMind project (CYD-C-2020003) and *(b)* the University of Zürich UZH.

References

1. Shilpa Kapse. Ethics in ai in machine learning. In *Handbook of Research on Machine Learning*, pages 3–24. Apple Academic Press, 2022.
2. K. Hu, Reuters. ChatGPT sets record for fastest-growing user base - analyst note, February 2023. https://www.reuters.com/technology/chatgpt-sets-record-fastest-growing-user-base-analyst-note-2023-02-01/, last accessed September 2, 2024.
3. Enrique Tomás Martínez Beltrán, Mario Quiles Pérez, Pedro Miguel Sánchez Sánchez, Sergio López Bernal, Gérôme Bovet, Manuel Gil Pérez, Gregorio Martínez Pérez, and Alberto Huertas Celdrán. Decentralized federated learning: Fundamentals, state of the art, frameworks, trends, and challenges. *IEEE Communications Surveys & Tutorials*, 2023.
4. Sawsan AbdulRahman, Hanine Tout, Hakima Ould-Slimane, Azzam Mourad, Chamseddine Talhi, and Mohsen Guizani. A survey on federated learning: The journey from centralized to distributed on-site learning and beyond. *IEEE Internet of Things Journal*, 8(7):5476–5497, 2020.
5. Li Li, Yuxi Fan, Mike Tse, and Kuo-Yi Lin. A review of applications in federated learning. *Computers & Industrial Engineering*, 149:106854, 2020.
6. Chao Feng, Alberto Huertas Celdrán, Jan von der Assen, Enrique Tomás Martínez Beltrán, Gérôme Bovet, and Burkhard Stiller. DART: A Solution for Decentralized Federated Learning Model Robustness Analysis. *Array*, 2024 (To appear).
7. Suzan Almutairi and Ahmed Barnawi. Federated learning vulnerabilities, threats and defenses: A systematic review and future directions. *Internet of Things*, 24:100947, 2023.
8. Priyanka Mary Mammen. Federated learning: Opportunities and challenges, 2021. https://arxiv.org/abs/2101.05428.
9. Bokolo Anthony Jr. Deployment of distributed ledger and decentralized technology for transition to smart industries. *Environment Systems and Decisions*, 43(2):298–319, 2023.
10. Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. http://www.bitcoin.org/bitcoin.pdf, May 2009.
11. Leslie Lamport, Robert Shostak, and Marshall Pease. *The Byzantine generals problem*, page 203–226. Association for Computing Machinery, New York, NY, USA, 2019.
12. Svein Ølnes, Jolien Ubacht, and Marijn Janssen. Blockchain in government: Benefits and implications of distributed ledger technology for information sharing. *Government Information Quarterly*, 34(3):355–364, 2017.
13. Tian Min and Wei Cai. Portrait of decentralized application users: an overview based on large-scale ethereum data. *CCF Transactions on Pervasive Computing and Interaction*, 4(2):124–141, 2022.
14. Zibin Zheng, Shaoan Xie, Hong-Ning Dai, Weili Chen, Xiangping Chen, Jian Weng, and Muhammad Imran. An overview on smart contracts: Challenges, advances and platforms. *Future Generation Computer Systems*, 105:475–491, 2020.
15. Sarah Bouraga. A taxonomy of blockchain consensus protocols: A survey and classification framework. *Expert Systems with Applications*, 168:114384, 2021.
16. Wei Cai, Zehua Wang, Jason B Ernst, Zhen Hong, Chen Feng, and Victor CM Leung. Decentralized applications: The blockchain-empowered software system. *IEEE access*, 6:53019–53033, 2018.
17. Ahmed S Almasoud, Farookh Khadeer Hussain, and Omar K Hussain. Smart contracts for blockchain-based reputation systems: A systematic literature review. *Journal of Network and Computer Applications*, 170:102814, 2020.
18. Zibin Zheng, Shaoan Xie, Hong-Ning Dai, Weili Chen, Xiangping Chen, Jian Weng, and Muhammad Imran. An overview on smart contracts: Challenges, advances and platforms. *Future Generation Computer Systems*, 105:475–491, 2020.
19. Vitalik Buterin et al. A next-generation smart contract and decentralized application platform. *white paper*, 3(37):2–1, 2014.
20. Yanru Chen, Jingpeng Li, Fan Wang, Kaifeng Yue, Yang Li, Bin Xing, Lei Zhang, and Liangyin Chen. Ds2pm: A data sharing privacy protection model based on blockchain and federated learning. *IEEE Internet of Things Journal*, 2021.
21. Youyang Qu, Shiva Raj Pokhrel, Sahil Garg, Longxiang Gao, and Yong Xiang. A blockchained federated learning framework for cognitive computing in industry 4.0 networks. *IEEE Transactions on Industrial Informatics*, 17(4):2964–2973, 2020.
22. Jiawen Kang, Zehui Xiong, Dusit Niyato, Yuze Zou, Yang Zhang, and Mohsen Guizani. Reliable federated learning for mobile networks. *IEEE Wireless Communications*, 27(2):72–80, 2020.

23. Yang Zhao, Jun Zhao, Linshan Jiang, Rui Tan, and Dusit Niyato. Mobile edge computing, blockchain and reputation-based crowdsourcing iot federated learning: A secure, decentralized and privacy-preserving system. *arXiv preprint arXiv:1906.10893*, pages 2327–4662, 2019.
24. Harsh Kasyap and Somanath Tripathy. Privacy-preserving and byzantine-robust federated learning framework using permissioned blockchain. *Expert Systems with Applications*, 238:122210, 2024.
25. Devrim Unal, Mohammad Hammoudeh, Muhammad Asif Khan, Abdelrahman Abuarqoub, Gregory Epiphaniou, and Ridha Hamila. Integration of federated machine learning and blockchain for the provision of secure big data analytics for internet of things. *Computers & Security*, 109:102393, 2021.
26. Jorge Castillo, Phillip Rieger, Hossein Fereidooni, Qian Chen, and Ahmad Sadeghi. Fledge: Ledger-based federated learning resilient to inference and backdoor attacks. In *Proceedings of the 39th Annual Computer Security Applications Conference*, pages 647–661, 2023.
27. Youssif Abuzied, Mohamed Ghanem, Fadi Dawoud, Habiba Gamal, Eslam Soliman, Hossam Sharara, and Tamer Elbatt. A privacy-preserving federated learning framework for blockchain networks. *Cluster Computing*, 27(4), 2023.
28. Umer Majeed and Choong Seon Hong. Flchain: Federated learning via mec-enabled blockchain network. in 2019 20th asia-pacific network operations and management symposium (apnoms). *IEEE, 1ś4*, 2019.
29. Harsh Kasyap, Arpan Manna, and Somanath Tripathy. An efficient blockchain assisted reputation aware decentralized federated learning framework. *IEEE Transactions on Network and Service Management*, 2022.
30. Ronghua Xu and Yu Chen. μdfl: A secure microchained decentralized federated learning fabric atop iot networks. *IEEE Transactions on Network and Service Management*, 19(3):2677–2688, 2022.
31. Enrique Tomás Martínez Beltrán, Ángel Luis Perales Gómez, Chao Feng, Pedro Miguel Sánchez Sánchez, Sergio López Bernal, Gérôme Bovet, Manuel Gil Pérez, Gregorio Martínez Pérez, and Alberto Huertas Celdrán. Fedstellar: A platform for decentralized federated learning. *arXiv e-prints*, pages arXiv–2306, 2023.
32. go-ethereum Authors. go-ethereum, 2024. https://geth.ethereum.org/ last accessed September 2, 2024.
33. enriquetomasmb. Nebula: A platform for decentralized federated learning, 2024. https://github.com/enriquetomasmb/nebula, last accessed September 2, 2024.