Trust and Resilience in Federated Learning Through Smart Contracts Enabled Decentralized Systems

Trust and Resilience in Federated Learning Through Smart Contracts Enabled Decentralized Systems

Lorenzo Cassano*, Jacopo D'Abramo*, Siraj Munir† Stefano Ferretti†*
*Department of Computer Science and Engineering, University of Bologna, Italy
†Department of Pure and Applied Sciences, University of Urbino Carlo Bo, Italy
{lorenzo.cassano2,jacopo.dabramo}@studio.unibo.it,
s.munir@campus.uniurb.it,stefano.ferretti@uniurb.it

***Abstract*—In this paper, we present a study of a Federated Learning (FL) system, based on the use of decentralized architectures to ensure trust and increase reliability. The system is based on the idea that the FL collaborators upload the (ciphered) model parameters on the Inter-Planetary File System (IPFS) and interact with a dedicated smart contract to track their behavior. Thank to this smart contract, the phases of parameter updates are managed efficiently, thereby strengthening data security. We have carried out an experimental study that exploits two different methods of weight aggregation, i.e., a classic averaging scheme and a federated proximal aggregation. The results confirm the feasibility of the proposal.**

***Index Terms*—Federated Learning, Blockchain, Decentralized Systems, Machine Learning, Smart Contracts**

I. Introduction

Federated Learning (FL) is a Machine Learning (ML) framework that enables multiple parties to collaboratively train a shared model without directly sharing their individual data [1]. FL holds the promise of enhancing data-driven learning models while safeguarding data owners' privacy. This makes it an appealing approach for developing ML models in several application domains. A prominent example is clinical diagnosis, where patient data privacy is critical, yet data aggregation from diverse sources is necessary. However, a significant challenge revolves around ensuring FL collaborators actively participate in the protocol while securely contributing their data. To this extent, recently several studies have explored the integration of blockchain technology within FL systems [2]–[6]. While existing research often emphasizes on participation incentives, traceability, and security aspects, there is an under-explored dimension that deserves attention, i.e., the impact of delays, number of collaborators, and collaborator failures on system performance and model training accuracy.

In our work, we investigate the performance of our FL system and the accuracy of ML training when one or more collaborators experience failures.

We propose a decentralized FL system that combines FL with two powerful components: the Inter-Planetary File System (IPFS) and smart contracts (executed over a permissioned Ethereum-like blockchain). Together, they create a tamper-proof storage mechanism for sharing encrypted model parameters. IPFS provides decentralized and fault-tolerant data storage, while smart contracts enforce rules and streamline interactions among participants [7]. The presence of a smart contract forces Collaborators to adhere to a protocol organized into distinct phases, ensuring orderly updates of model parameters. Additionally, the smart contract automates compliance checks, maintaining the integrity of the FL process [6].

In the paper, we present the overall decentralized system and we perform an experimental evaluation under varying failure scenarios among Collaborators. As concerns the FL model assessment, two weight update methods are used, *FedAvg* and *FedProx* [8]. We investigate the FL system's performance in terms of classification accuracy, using a dataset coming from the healthcare image classification domain, i.e., brain tumor image dataset. Furthermore, we measure the gas consumption of the smart contract, a critical aspect in blockchain-based FL systems. Finally, we also evaluate the performance of the data retrieval and update, by both FL Manager and Collaborators, from/to IPFS. Results confirm the viability of using decentralized systems in Federated Learning.

The remainder of this paper is organized as follows. Section II describes the proposed framework. Section III describes the methodology, while Section IV outlines the experimental results. Section V provides some concluding remarks.

II. System Architecture

Our system architecture comprises four key actors. The first two are the classic actors involved in a typical FL system, while the last two ones are those of typical decentralized systems [7]:

- **FL Manager**: The Manager acts as the aggregator for the classification model parameters used in FL. It triggers requests, collects ML parameters from other Collaborators, and monitors parameter updates.
- **FL Collaborators**: these are the nodes participating to FL model training. They retrieve data, locally train their models, and upload the trained parameters to the system.
- **Permissioned Blockchain**: This blockchain executes a Federated Learning Smart Contract (FLSC). The FLSC autonomously regulates parameter exchanges among participants, ensuring adherence to the protocol. Additionally, the use of blockchain digests prevents tampering with parameters, enhancing security [9], [10].
- **Decentralized File Storage**: The Inter-Planetary File System (IPFS) stores the ML model parameters.

A. System Interaction Overview

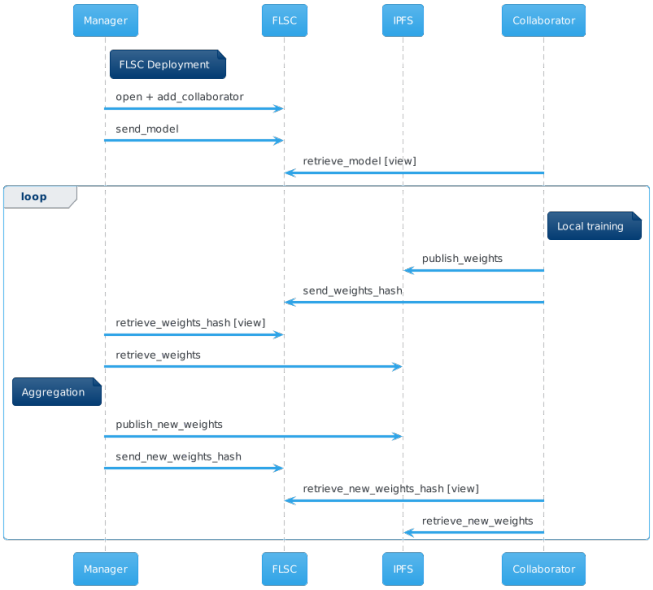


Fig. 1: System diagram

The diagram of the interactions among system components is shown in Figure 1. The protocol works as follows:

1) The Manager deploys and initiates the Federated Learning Smart Contract (FLSC), inviting multiple Collaborators (for simplicity, only one is depicted in the figure).
2) The Manager publishes the model details required for compilation within FLSC.
3) Collaborators retrieve and adopt the model from the FL Smart Contract.

FL Loop

4) Each Collaborator trains its local model using its dataset. By keeping data decentralized, FL avoids the need for centralizing sensitive information. This brings positive aspects possibly related to enhanced privacy, data protection, regulatory compliance, and scalability.
5) Periodically, Collaborators publish updated local parameters on IPFS and send their parameter hashes (along with retrieval information) to FLSC. Parameters are encrypted sequentially using the Collaborator's private key and the Manager's public key. This ensures confidentiality and verifiability — only the Manager can read the data, and their origin is guaranteed.
6) The Manager decrypts these parameters using its private key and the Collaborator's public key. The Manager further validates parameters by comparing their digest to the hashed value stored in the smart contract.
7) Based on an aggregation function, the Manager updates the ML model parameters, considering results obtained during training.
8) The Manager securely uploads the updated ML model parameters to IPFS, using the same encryption approach as previously described. These parameters represent the refined knowledge gained during the training process.
9) The Manager sends the digests (hashes) of these parameters to the smart contract. These digests serve as verifiable references for the stored parameters.
10) Collaborators retrieve the updated model parameters from IPFS. They decrypt the parameters using their private keys. Collaborators then verify the validity of these parameters through the FL Smart Contract, following the same process as before. This validation ensures that the parameters have not been tampered with and align with the agreed-upon protocol. Once validated, Collaborators incorporate these updated model parameters into their local training during the subsequent iteration.

In our current implementation, Collaborators receive no rewards for parameter updates, assuming protocol compliance. However, our framework allows for alternative security-enhancing or incentive-based approaches. For example, participants might be automatically rewarded based on their contributions [2], [3], [11], [12].

B. The Federated Learning Smart Contract

The FLSC has been implemented in Solidity. The contract structure incorporates methods that align with the state phases depicted in Figure 1, i.e, all messages involving the FLSC. It establishes four states for the FL process: OPEN, START, LEARNING, and CLOSE. These states guide the progression of the FL process, ensuring it adheres to predefined rules. This structured approach forces Collaborators to engage with the FLSC only when necessary. Furthermore, the phase definitions facilitate synchronization among all parties involved in FL, effectively addressing issues related to node failures. This implies that if a node fails or is late in transmitting the hash of its weights, the training process is not hindered, and weight updates proceed without considering the contribution of that particular node.

Collaborators are added to the contract during the initialization phase, allowing only authorized users to participate. Collaborators can be added by the contract owner, and their interactions are monitored throughout the FL process. In Figure 1, we highlight the view functions, which correspond to methods within the smart contract that do not alter the contract state. These functions do not involve gas consumption for their execution and provide read-only access to specific data or parameters. The contract includes security measures to restrict unauthorized access and ensure that actions are carried out only by authorized users. During each round of the FL, Collaborators can perform specific actions only once, and the contract owner (Manager) has control over the FL process. Finally, the contract emits events to signal state transitions during the FL process, providing transparency and auditability.

C. On the Security and Reliability of the Protocol

The protocol is designed to handle the possibility of Collaborators failures. However, it clearly assumes that the Manager cannot fail. The Manager is responsible for coordinating all

activities, including sending messages to the FLSC, computing the updated weights, uploading the new values to IPFS, and notifying the Collaborators via smart contract calls, which in turn will emit events that the Collaborators can wait for.

The security of the protocol also relies on the Manager. Collaborators encrypt their weights using the Manager's public key and their own private key. This ensures verifiability (i.e., anyone can verify that the message was generated by a specific Collaborator) and confidentiality (i.e., only the Manager can decrypt the values).

These aspects make the Manager a possible single point of failure for the system. If the Manager fails, no one can carry out the training phase, as no one can invoke the appropriate functions of the FLSC smart contract or decrypt the values.

This limitation can be addressed by adopting one of the classic consensus protocols in an asynchronous network for managing a primary node election [13]. Any problems related to data encryption could be solved by using mechanisms that facilitate the use of encrypted decentralized file storage while enabling data sharing, such as key re-distribution techniques and multi-party computation [12].

III. Experimental Evaluation

We performed an experimental analysis to evaluate, on one side, the performance of the FL system, based on different aggregation methods and system setting. We also varied the amount of total nodes involved in the FL, with the possibility of having node failures during the training process, so as to assess how these node failures impact the overall performance. On the other hand, we measured the gas fees of the smart contract system, based on the amount of involved nodes.

A. The Federated Learning Model and Weight Update Methods

Collaborators used the same ML approach for classification, i.e., a three-layered, two-dimensional Convolutional Neural Network (CNN) model, coupled with a final fully connected layer at the end for the final classification of each image. The models are trained with the same setup: an Adam optimizer, a learning rate set to 10e-3, batch size equal to 32, softmax as the last activation function.

In this work, we used two specific FL weight update methods, inspired by [8]. In few words, we will denote with *FedAvg* the classic approach that assumes that all Collaborators contribute equally and that the dataset is evenly distributed across all nodes. Thus, when weights need to be updated at the end of a loop iteration, the novel weight is just the average value of weights received from the Collaborators. In this case, the classic cross-entropy is used as the loss function.

Federated Proximal (*FexProx*), instead, removes the uniformity assumption. Proximal optimization involves adding a regularization term to the objective function to encourage specific properties (e.g., sparsity). At the end of the FL iteration happened at timestep t, weights are updated locally in order to minimize a loss function of this form

$$\arg\min_w F(w) + \frac{\mu}{2}\|w - w^t\|_2^2,$$

where $F(w)$ represents the local loss function, w denotes the local weights to be identified, and w^t signifies the global weights sent by the Manager at the beginning of timestep t. The Manager collects these weights coming from Collaborators, and then computes a novel version of these weights w^{t+1} by taking the average of the received values.

During our experiments, we tried with different values for the μ hyper-parameter, finding that after the tuning the best results were obtained with $\mu = 0.001$. Thus, we show results obtained according to this setting.

B. Brain Tumor Dataset

The dataset we employed during the tests is Brain Tumor MRI Dataset: https://www.kaggle.com/datasets/iashiqul/brain-tumor-mri-image-classification. This dataset contains 7,022 human brain MRI images classified into four classes: glioma, meningioma, no tumor, and pituitary.

IV. Results

The implemented system involves various technologies, i.e., i) FL techniques for the decentralized analysis of datasets, which are partitioned across multiple sources; ii) blockchain technologies, particularly the FLSC, to coordinate the training phases, iii) IPFS, used for the decentralized yet secure exchange of information, thanks to data encryption. To evaluate this type of system, it is thus necessary to consider its various aspects. In this section, we show results obtained by using the FL system on the mentioned dataset related to image classification in healthcare, which is a typical use-case application for FL. Then, we will show the performance of the FLSC used to orchestrate the interaction among FL nodes [14]–[16]. Finally, we show also experimental results obtained for the weights retrieval and transmission to IPFS.

A. FL performance

Table I shows the averaged accuracy and F1 score obtained by the two FL aggregation techniques, against a centralized approach, i.e., the one that uses a classic ML, where the whole dataset is stored at a central node. In this case, er show results obtained in the best configuration, i.e., when the amount of employed Collaborators was equal to 5, with no failing nodes, together with the Manager (i.e., 6 nodes in total). Without any surprise, best results are obtained with the centralized approach. Having all the instances in the same data-store eases the training, and this leads to best results. However, we discussed already that in certain scenarios this is not possible. Thus, the centralized approach should be taken as an upper bound for the FL methods. In this respect, it is interesting to observe that the FL schemes perform quite well, especially FedAvg that is only 0.02 scores below centralized.

TABLE I: Accuracy Results. Number of Collaborators = 5

Agg. Method	Accuracy	F1_score
Centralized	0.98	0.98
FedAvg	0.96	0.96
FedProx	0.96	0.95

TABLE II: Classification results for FedAvg and FedProx. Number of Collaborators = 5. Accuracy FedAvg=0.96, FedProx=0.96

	Precision	Recall	F1-score	Support
Glioma	0.99 / 0.94	0.90 / 0.94	0.94 / 0.94	299
Meningioma	0.89 / 0.92	0.95 / 0.90	0.92 / 0.91	301
Notumor	0.98 / 0.98	0.99 / 0.99	0.99 / 0.98	381
Pituitary	0.98 / 0.98	0.99 / 0.99	0.99 / 0.99	300

Table II shows FL results related to each class. In each cell, the first value corresponds to FedAvg and the second value corresponds to FedProx. Both methods seem to perform well, with FedAvg which is slightly superior to FedProx in terms of F1 score.

Figure 2 shows the ROC curves for both aggregation schemes, FedAvg (left chart) and FedProx (right chart). Each chart shows a different curve for each class of the dataset. The AUC values are reported as well in the legend, for each class. It is possible to appreciate how both schemes are able to obtain very good performance.

Figure3 shows the accuracy and F1 score, for both the aggregation methods, when different numbers of Collaborators are used. We can notice that FedAvg remains mostly stable, while FedProx has a decrement when the number of collaborators is equal to 10. This is a result that needs some further evaluation, but that is probably due to the limited size of the dataset. In fact, when we increase the number of Collaborators, the dataset is split into multiple parts and, evidently, not all Collaborators are able to improve the overall performance.

Figure 4 reports both the accuracy and F1 score as a function of the number of failing nodes, when the total amount of involved nodes (active and inactive) was equal to 20 Collaborators (plus the Manager). Both aggregation methods, FedProx and FedAvg, are reported. Accuracy and F1 scores of each method change as the number of failing nodes increases. In general, the trend is that, as expected, as the number of failing nodes increases, the performance decreases. This is due to the fact that part of the whole dataset was distributed among nodes (also failed ones) and the loss of the contribution of certain collaborators affects the training. It is worth pointing out that we obtained similar results for other settings with different number of nodes in the system (not shown here for the sake of space limits).

Overall, results show that for this dataset the two compared approaches behave quite similarly in terms of performance, hence confirming the effectiveness of using FL in the healthcare sector. They highlight the influence of node failures and the importance of being able to transparently assess the correct execution of the protocol.

B. Federated Learning Smart Contract Gas Consumption

In this section, we show the gas consumption analysis for the smart contract that governs the interactions in the FL system. Due to space limitations, we will show only results related to system settings with no node failures.

TABLE III: Collaborators fee statistics for *send_weights_hash()* function

	gas consumed
mean	1390385.00
std	88532.56
median	1376425.00
min	1285740.00
25%	1331082.50
50%	1376425.00
75%	1426507.50
max	1559830.00

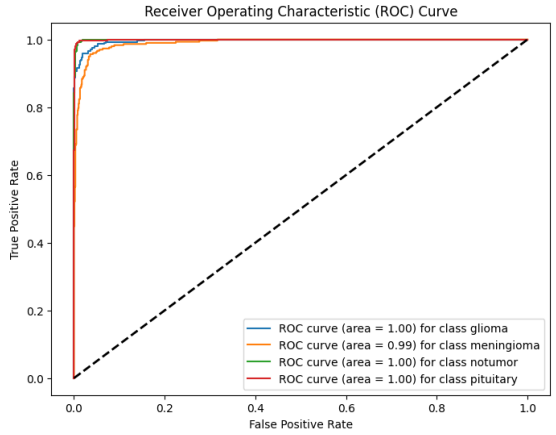
Table III shows the gas fee consumption for the *send_weights_hash()* , i.e., the only non-view function in FLSC called by Collaborators. Results show a low level of dispersion, since the standard deviation is relatively small compared to the mean. This outcome was indeed expected, since with this function a Collaborator registers in the smart contract a hash of the weights it just uploaded to IPFS. Thus, it basically sends a fixed length datum.

Figure 5 shows gas fees for different methods called by the Manager as a function of the number of Collaborators involved in the FL system. Each line represents a different method. From the chart, it is possible to observe how the gas fees increase linearly with the amount of nodes in the system, with the exception of *send_model*, that clearly does not depend on this variable. The cost for the deployment of the contract was 2151147 gas. These results confirm the viability of the approach, especially if we assume that the number of Collaborators involved in a FL system is limited and does not require continuous updates to the set of participants. In this latter scenario, maybe some optimization techniques might be necessary.

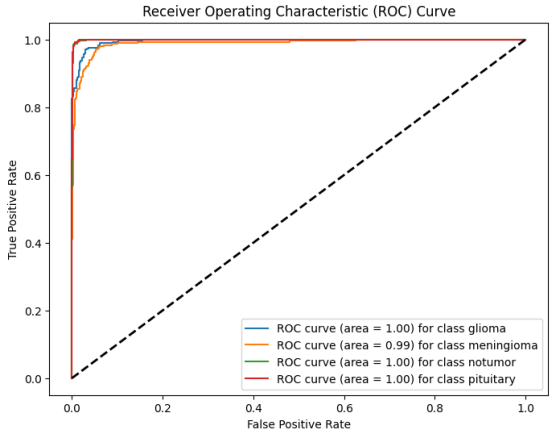
C. IPFS Delays

Figures 6 and 7 show the average times required for the Manager and the Collaborators respectively, to perform essential actions related to updating weights on the Inter-Planetary File System (IPFS). Specifically, we examine the time taken to send (referred to as “Add Operation”) and retrieve (referred to as “Cat Operation”) the updated weights. The figures display both the averages and standard deviations of the measurements collected. Each individual configuration was repeated 10 times. In each run, the total number of participating nodes was the sum of the Manager and the Collaborators, the number of which is indicated in the chart. The execution spanned 10 rounds of the FL protocol. In this context, since the metrics measured were related to the transmission of weights to IPFS, all nodes were run on a single server with the following technical specifications: Intel(R) Core(TM) i7-8565U CPU (1.80 GHz - 1.99 GHz), 16.0 GB RAM, and a symmetrical bandwidth connection of 433/433 Mbps.

Figures show that, as expected, delays are comparable for both the Manager and Collaborators. This consistency arises since the amount of data to be sent or received is the same, i.e., the data comprises a set of numerical values representing the weights of the employed Convolutional Neural Network



(a) FedAvg



(b) FedProx

Fig. 2: ROC curves for Brain Tumor dataset. Number of Collaborators = 5

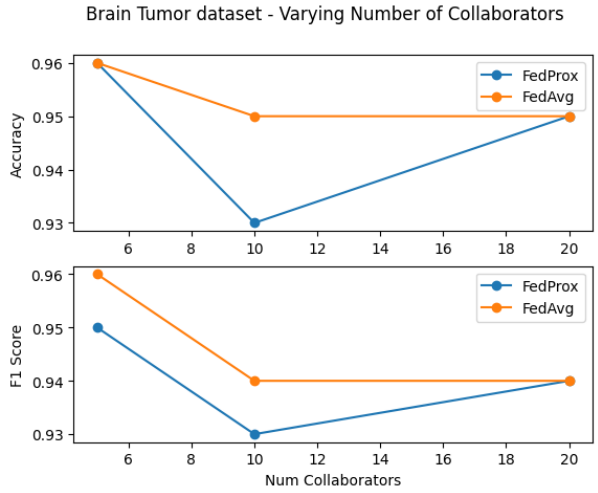


Fig. 3: Accuracy and F1 score performances with a varying number of Collaborators

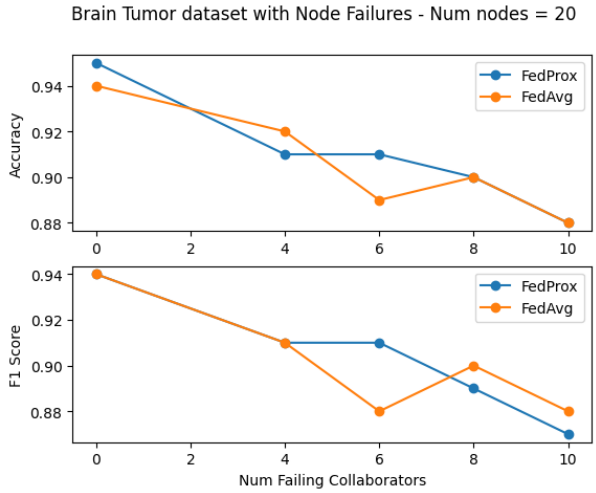


Fig. 4: Accuracy and F1 score performances with node failures

(CNN). Moreover, the time to retrieve the novel weights was slightly higher than sending them. Finally, average times do not significantly vary based on the amount of involved Collaborators, since the amount of data to be retrieved for each Collaborator remains constant. There are differences in average values and standard deviations, that are emphasized in the charts only because of the limited scale of the y-axis. But in the end, results do not show significant differences.

In summary, however, our measurements confirm that there is a non-negligible time overhead associated with coordinating FL activities. Indeed, at the end of each round every Collaborator has to upload its weights to IPFS, compute and send the related hash value to the smart contract, and wait for an smart contract event, so as to retrieve the hash from the smart contract and the weights from IPFS. In the middle of this round step, the Manager has to retrieve all the hashes related to each Collaborator from the smart contract, retrieve all the weights of all the Collaborators, compute the average and upload the updated weights on IPFS. Given the measured times, depending on the number of Collaborators this upload phase requires some tens of seconds.

It is important to notice that we observed consistent performance on IPFS even in the event of node failures. Clearly enough, this process remains independent of other nodes, relying solely on the interaction with IPFS. For this reason and for the sake of brevity, we thus omit these results.

V. Conclusions

In this study, we presented a Federated Learning (FL) system that is based on the use of blockchain technology, aiming for a secure and coordinated approach to data management. The design of the framework is centered around data privacy protection, where it exchanges encrypted model parameters instead of sensitive data. This system integrates the Inter-Planetary File System (IPFS) and smart contracts to

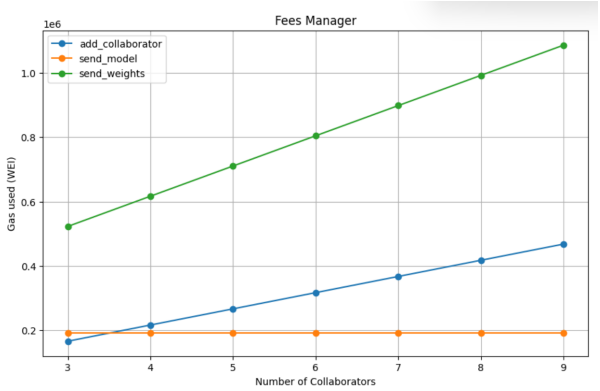


Fig. 5: Manager: Gas Fees Incurred

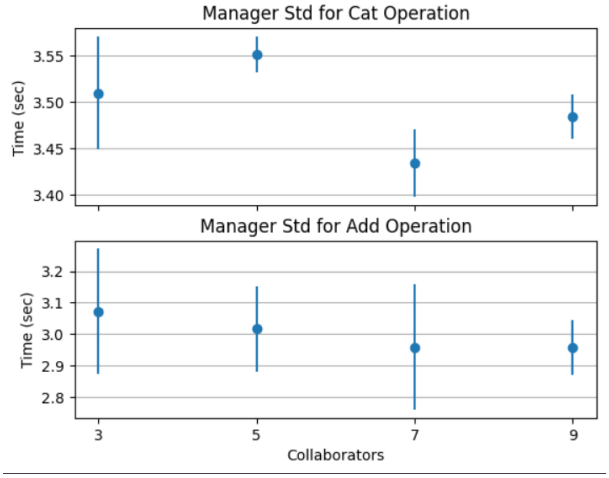


Fig. 6: Manager: IPFS delays based on the number of Collaborators

create a secure, unalterable data storage infrastructure, thereby enhancing data security during FL operations. The viability of our proposal is confirmed by the results from our validation and tests.

The complete code for the project can be accessed here: https://github.com/LorenzoCassano/Blockchain-FederatedLearning/tree/main.

Acknowledgements

This work has been partially funded by the European Union - NextGenerationEU within the framework of PNRR Mission 4 - Component 2 - Investment 1.1 under the Italian Ministry of University and Research (MUR) programme "PRIN 2022" - grant number 2022N2NH42 - SmartShires - CUP: H53D23003570006

References

[1] J. Wen, Z. Zhang, Y. Lan, Z. Cui, J. Cai, and W. Zhang, "A survey on federated learning: challenges and applications," *International Journal of Machine Learning and Cybernetics*, vol. 14, no. 2, pp. 513–535, 2023.

[2] M. R. Behera, S. Upadhyay, and S. Shetty, "Federated learning using smart contracts on blockchains, based on reward driven approach," *ArXiv*, vol. abs/2107.10243, 2021.

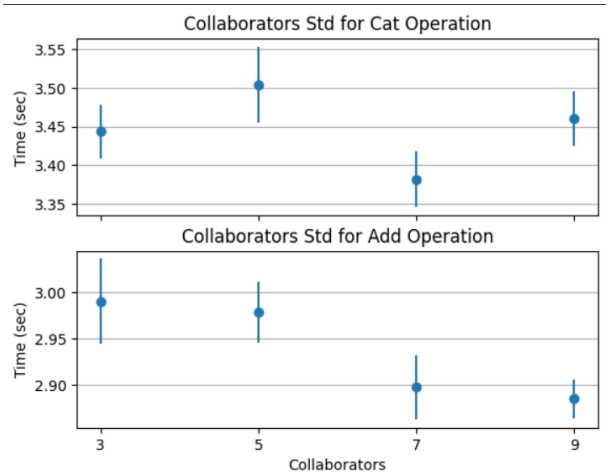


Fig. 7: Collaborators: IPFS delays based on the number of Collaborators

[3] A. Qammar, A. Karim, H. Ning, and J. Ding, "Securing federated learning with blockchain: a systematic literature review," *Artificial Intelligence Review*, vol. 56, pp. 3951–3985, 2023.

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

[5] X. Bao, C. Su, Y. Xiong, W. Huang, and Y. Hu, "Flchain: A blockchain for auditable federated learning with trust and incentive," in *5th Int. Conf. on Big Data Computing and Communications (BIGCOM)*, 2019, pp. 151–159.

[6] F. Imboccioli, G. Cialone, and S. Ferretti, "Decentralization of learning and trust in healthcare: Blockchain-driven federated learning for alzheimer's mri image classification," in *Proc. of the IEEE PerCom Workshops*. Biarritz, France: IEEE, March 2024.

[7] M. Zichichi, S. Ferretti, and G. D'Angelo, "On the efficiency of decentralized file storage for personal information management systems," in *Proceedings of the IEEE Symposium on Computers and Communications*. IEEE, July 2020.

[8] T. L. et al., "Federated optimization in heterogeneous networks," in *Proc. of Machine Learning and Systems 2020, MLSys 2020, March 2020*. mlsys.org, 2020. [Online]. Available: https://proceedings.mlsys.org/book/316.pdf

[9] G. Bigini, M. Zichichi, E. Lattanzi, S. Ferretti, and G. D'Angelo, "Decentralized health data distribution: A dlt-based architecture for data protection," 2022, Conference paper, p. 97 – 104.

[10] L. Serena, G. D'Angelo, and S. Ferretti, "Security analysis of distributed ledgers and blockchains through agent-based simulation," *Simulation Modelling Practice and Theory*, vol. 114, 2022.

[11] M. Zichichi, S. Ferretti, G. D'Angelo, and V. Rodríguez-Doncel, "Data governance through a multi-dlt architecture in view of the gdpr," *Cluster Computing*, pp. 1–32, 2022.

[12] F. Barbàra, M. Zichichi, S. Ferretti, and C. Schifanella, "Dlt-based personal data access control with key-redistribution," 2023, p. 166 – 173.

[13] M. Pease, R. Shostak, and L. Lamport, "Reaching agreement in the presence of faults," *J. ACM*, vol. 27, no. 2, p. 228–234, apr 1980. [Online]. Available: https://doi.org/10.1145/322186.322188

[14] K. K. Coelho, M. Nogueira, A. B. Vieira, E. F. Silva, and J. A. M. Nacif, "A survey on federated learning for security and privacy in healthcare applications," *Computer Communications*, vol. 207, pp. 113–127, 2023.

[15] R. S. Antunes, C. André da Costa, A. Küderle, I. A. Yari, and B. Eskofier, "Federated learning for healthcare: Systematic review and architecture proposal," *ACM Trans. Intell. Syst. Technol.*, vol. 13, 2022.

[16] T.S. Brisimi et al., "Federated learning of predictive models from federated electronic health records," *International Journal of Medical Informatics*, vol. 112, pp. 59–67, 2018.