Online-Journals.org

iJOE International Journal of Online and Biomedical Engineering

iJOE | eISSN: 2626-8493 | Vol. 20 No. 11 (2024) | OPEN ACCESS

https://doi.org/10.3991/ijoe.v20i11.49949

PAPER

Federated-Learning Intrusion Detection System Based Blockchain Technology

Ahmed Almaghthawi[1](✉), Ebrahim A. A. Ghaleb[2], Nur Arifin Akbar[2], Layla Asiri[3], Meaad Alrehaili[4], Askar Altalidi[5]

[1]Department of Computer Science, College of Science and Art at Mahayil, King Khalid University, Abha, Saudi Arabia

[2]Departement of Computer and Information Sciences, University Teknologi PETRONAS, Seri Iskandar, Malaysia

[3]Departement of Computer Information System, Applied College at Mahayil, King Khalid University, Abha, Saudi Arabia

[4]Department of Computer Sciences and Artificial Intelligence, Collège of Computer Science, and Engineering, University of Jeddah, Jeddah, Saudi Arabia

[5]Department of Information Science, College of Science & Art at Mahayil, King Khalid University, Abha, Saudi Arabia

aalmaghthwi@kku.edu.sa

ABSTRACT

This study presents the implementation of a blockchain-based federated-learning (FL) intrusion detection system. This approach utilizes machine learning (ML) instead of traditional signature-based methods, enabling the system to detect new attack types. The FL technique ensures the privacy of sensitive data while still utilizing the large amounts of data distributed across client devices. To achieve this, we employed the federated averaging method and incorporated a custom preprocessing stage for data standardization. The use of blockchain technology in combination with FL created a fully decentralized and open learning system capable of overcoming new security challenges.

KEYWORDS

machine learning (ML), federated learning (FL), blockchain, intrusion detection security

1 INTRODUCTION

With the exponential growth in computing power, machine learning (ML) has become a ubiquitous technique applied to virtually any problem that meets the criteria of having the required amount of data [1]. In many ML applications, vast quantities of user data are collected to train the ML model and produce highly accurate results [2]. However, the conventional approach of storing user data on servers has inherent drawbacks [3, 4]. Specifically, collecting a significant amount of data requires considerable computational and network resources, while the collected data may contain sensitive user information that users may be reluctant to share [5].

Most intrusion detection systems rely on well-known signatures of attack vectors to classify malicious users based on their actions. While this approach may work well for attack vectors that have been previously investigated by cybersecurity experts, it cannot accurately identify new attack vectors, such as zero-day attacks [6, 7]. To overcome this challenge, statistics- or ML-based approaches are utilized, particularly anomaly-based intrusion detection systems [8]. However, ML-based IDSs require training data from users, raising privacy concerns.

Almaghthawi, A., Ghaleb, E.A.A., Akbar, N.A., Asiri, L., Alrehaili, M., Altalidi, A. (2024). Federated-Learning Intrusion Detection System Based Blockchain Technology. *International Journal of Online and Biomedical Engineering (iJOE)*, 20(11), pp. 16–30. https://doi.org/10.3991/ijoe.v20i11.49949

Article submitted 2024-03-02. Revision uploaded 2024-05-26. Final acceptance 2024-05-26.

Federated learning (FL) is a novel ML model developed to tackle privacy-related concerns. It is a technique where multiple clients, each with their own private data, work together to train a ML model [9]. Horizontal FL involves different samples with the same feature space and characteristics for each client, while vertical FL entails clients using the same samples with different feature spaces during local training [9].

Despite their potential to solve the aforementioned privacy-related issues, FL models still rely on a centralized server for storing the learning results, making them vulnerable to single-point-of-failure attacks [10]. To tackle this issue, a blockchain-based FL system has been developed, which leverages the distributed structure of blockchain to enhance the system's security. This approach enables the development of a more robust and accurate IDS that can detect new and emerging threats, such as zero-day attacks, while maintaining user privacy. One specific application of FL in IDSs is the use of blockchain technology [11], which can improve the accuracy and speed of IDSs by using blockchain technology to secure the data and the model parameters. Finally, our study aims to bridge this gap by implementing an FL intrusion detection system based on blockchain technology.

This paper is organized into five main sections. Section 1 introduces the background and motivation behind the implementation of an FL intrusion detection system based on blockchain technology. Section 2 presents a literature review of the relevant topics, including FL, intrusion detection systems, and blockchain technology. In Section 3, we describe the methodology used in our system, including data preprocessing, model architecture, and the training process. Section 4 presents the experimental results, including the performance evaluation of the proposed system on a publicly available dataset and the limitations of the proposed approach, while Section 5 provides a conclusion.

2 RELATED WORK

Intrusion detection systems (IDS) are critical components of modern cybersecurity strategies. They can be either signature-based, which can detect known attacks, or ML-based, which can detect new types of attacks. However, traditional IDS face several challenges, such as scalability, privacy, and the need for centralized data storage. In recent years, FL has emerged as a promising solution to these challenges, allowing multiple parties to train ML models collaboratively without sharing sensitive data. Additionally, blockchain technology provides a decentralized and transparent platform for securely storing and sharing data. Combining FL and blockchain can lead to more secure and private intrusion detection systems.

Performance comparisons of different FL algorithms were conducted in [12], and federated averaging achieved the highest accuracy. However, it has also been demonstrated that FL faces challenges when the data is non-IID (non-independent and identically distributed). To address this issue, [13] introduces a hierarchical clustering approach.

A fairly comprehensive literature survey about FL for IDS is provided in [13]. A more general survey is given in [14], concerning IDS using machine learning. These papers also offer a wealth of background information on IDS, which we will omit for the sake of brevity.

When the training data is scarce, anomaly detection systems may produce numerous false positives and demonstrate poor overall performance. FL allows us to utilize the private data stored on various devices without directly sharing the data.

For instance, in the collaborative IDS discussed in [15], FL is used alongside a semi-supervised learning algorithm to address data scarcity problems. Likewise, in [16], the privacy-preserving features of FL are utilized to safeguard sensitive data while training the model.

Furthermore, due to its simplicity, FL offers an elegant and flexible framework for building unique ML applications. A single deep neural network is trained for multiple IDS tasks in [1] using FL, which even surpassed some models dedicated to a single task. An example of using blockchain in combination with ML is given in [17], which presents a deep blockchain framework for collaborative intrusion detection systems.

3 METHODS

3.1 Machine learning model

In this study, the utilization of neural networks was chosen due to their inherent flexibility, allowing for easy customization through a simple yet expressive configuration system. The architecture of the neural network can be tailored to specific requirements, enabling the evaluation of algorithm accuracy under various configurations. Regardless of the specific configuration, the last layer of the neural network consists of L neurons, which provide confidence values for L classes. The performance of the algorithm is assessed by comparing the outputs against the ground truth using the cross-entropy loss function.

Horizontal FL was employed in this study, where all clients possess different training samples while sharing the same feature space. This approach was adopted because it represents a more common use case in practical applications. By utilizing horizontal FL, the study aims to address real-world scenarios where clients possess diverse datasets while focusing on the shared features.

The core concept of FL is introduced in a highly influential paper by [18, 19], which has had a significant impact on the field. The authors suggest a learning approach where clients conduct local updates to the model, and the server combines these updates through federated averaging to conclude each round. This procedure is akin to stochastic gradient descent, a commonly used optimization technique in ML. By embracing the federated averaging method, the study aligns with this established approach while expanding it to the blockchain-based FL environment.

By leveraging the flexibility of neural networks, employing horizontal federated learning, and incorporating the fundamental principles of federated averaging, this study aims to contribute to the field of FL and further advance its practical applications. The influence of reference [8] has provided a solid foundation for this research, guiding the implementation and evaluation of the proposed methodology.

Let's w_t^k denote the model parameters for client k at time t. Similarly, w_t denotes the global model parameters. At each round, the clients perform a local update on their weights individually:

$$w_{t+1}^k \leftarrow w_t^k - \eta \nabla l(w_t^k) \tag{1}$$

Where η is the learning rate, l represents the loss function (computed on the local dataset), and ∇l is its gradient w.r.t, model parameters. Since the updates performed by each client are independent of each other, they can work in parallel.

Additionally, each client has the flexibility to execute this parameter update step multiple times. A single iteration through the local dataset is referred to as a local epoch.

Instead of performing local updates with all clients at each round, the authors of [18] introduce another hyperparameter, C, and only conduct local updates with a C-fraction of clients at each round. C acts somewhat like a global batch size in this scenario; C = 1 implies that all clients perform updates, ensuring that all the data is utilized at each round, while smaller C values decrease this amount. The impact of adjusting C will be assessed in the results section.

After the local updates are completed, the server averages them to update the global model parameters:

$$w_{t+1} = \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^{k} \tag{2}$$

Where n_k is the amount of local data in kth client, and n is the total, i.e., $n = \sum_{k=1}^{K} n_k$ represents the total number of clients. This concludes a single round of FL. In the following round, the clients will fetch the averaged global model and perform gradient descent on it again.

We chose the PyTorch and NumPy libraries to implement this ML model in Python due to my pre-existing familiarity with these tools. To read the dataset, we used the Pandas library. Scikit-learn is also utilized for some preprocessing steps (e.g., converting categorical data to one-hot vectors).

3.2 Data standardization

With data standardization, we refer to the preprocessing step in which we eliminate the varying means and standard deviations of different features in the dataset. This process is particularly important when the features have wildly different scales, e.g., one feature has a mean of 10000 and the other 0.001. In fact, data standardization is deemed "necessary" when a non-linear activation function is utilized [20]. Intuitively, non-standardized values are less likely to trigger the non-linear response of an activation function; e.g., ReLU is linear for all positive numbers.

In a regular ML application, we usually remove the mean and divide the result by the standard deviation for each feature. In literature, this is commonly known in literature, this is commonly known as z-score:

$$z = \frac{x - \mu}{3} \tag{3}$$

Unfortunately, data standardization is not studied in the FL paper [18], presumably because it mainly targets image data, which is already normalized. Likewise, we failed to find any research about this issue in the literature. However, standardization is a crucial step in this application, as we will demonstrate in the results section. Moreover, we aimed to make this project more general-purpose by not making any assumptions about the data.

My solution to data standardization in FL is based on the local update and general averaging steps explained in the previous section. Before the training process starts, the server and clients participate in two sequential preprocessing stages for data standardization, one for the mean and the other for the standard deviation.

Similar to the local update step, the clients report their local means to the server first, μ_k for each client k, alongside their data size n_k. After that, the server computer calculates the overall mean of the formula.

$$\mu = \frac{\sum_{k=1}^{K} n_k \mu_k}{n} \tag{4}$$

This computed μ will be used by all clients to standardize their data. A similar process is carried out for the standard deviation. An important detail is that each client uses the global μ when computing their local standard deviation instead of the local μ_k.

$$\sigma_k = \sqrt{\frac{\sum_{i-1}^{n_k} (x_i - \mu)^2}{n_k}} \tag{5}$$

$$\sigma = \sqrt{\frac{\sum_{k-1}^{K} n_k \sigma_k^2}{n}} \tag{6}$$

Also note that these two standardization steps must be taken for each feature. However, thanks to vectorized operations, this does not require a separate pass for each feature. In practice, μ, μ_k, σ, and σ_k are vectors of values computed for all features, e.g., $\mu_k = [\mu_{k,1}, \mu_{k,2}]$.

Impact on privacy. One of the primary motivations for FL is to preserve the privacy of sensitive data, which is achieved by performing the updates locally and averaging them. A stochastic gradient descent update tells very little about the training data. However, the mean and standard deviation preprocessing steps that we introduced require the clients to send the means and standard deviations of their local data, which arguably exposes more sensitive information compared to gradient descent updates.

Similar to how only a fraction of clients participate in each round of gradient descent, we can limit the participation to volunteers in these two preprocessing stages. We will call this fraction care. With this approach, μ and σ values we compute will not be exactly correct, but they will provide a reasonable estimate, which is adequate in most cases.

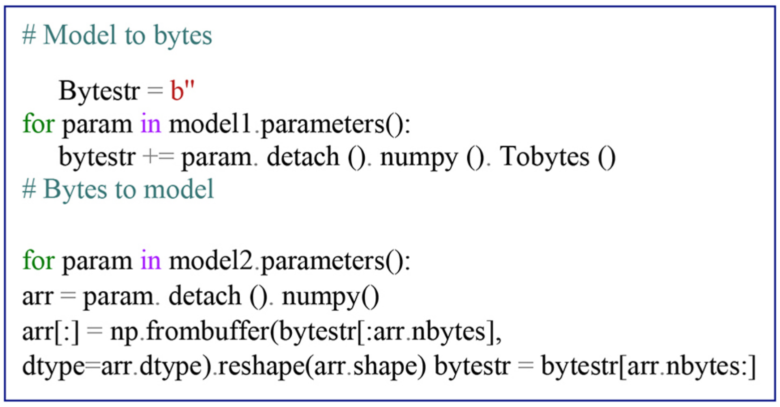
Furthermore, in real life, the dataset sizes of different clients are unlikely to be the same. When a client hoards a large amount of data, they will have less privacy concerns about sharing the mean and standard deviation information because these values tell a lot less about the individual data samples when the dataset size is large. This is quite convenient for us since the μ_k and σ_k values shared by these clients also have a larger impact on the global μ and σ. An alternative solution is simply to ditch the standardization for the sake of privacy. We will investigate the effect of this choice in the results section.

3.3 Blockchain

In this study, the primary blockchain platform of choice is Ethereum due to its popularity and extensive tooling support. The Solidity programming language was chosen for the development of the smart contract. To interact with the Ethereum blockchain, a Python interface was utilized, which is provided by the web3.py library. Given that running this project on the actual Ethereum network would be quite expensive, the eth-tester tool suite was employed to simulate the blockchain network.

To ensure a modular implementation of this blockchain platform, the EthPlatform module was developed to facilitate communication with the Ethereum ecosystem. This structure allows for easy replacement of the EthPlatform module and the provision of a DummyPlatform module, which can emulate the functions of the former without interacting with an actual blockchain network. This feature not only helps to bypass the runtime performance cost of blockchain but also benefits users who seek to test the FL component without installing any blockchain libraries.

In solidity, the model's state is represented as bytes. This is accomplished by aggregating all the model parameters in IEEE 754 floating-point representation to obtain the corresponding byte representation. This byte data can then be used by interpreting the byte buffer as an array with the appropriate data type and shape. Figure 1 demonstrates this process in Python.



```
# Model to bytes

    Bytestr = b''
for param in model1.parameters():
    bytestr += param. detach (). numpy (). Tobytes ()
# Bytes to model

for param in model2.parameters():
arr = param. detach (). numpy()
arr[:] = np.frombuffer(bytestr[:arr.nbytes],
dtype=arr.dtype).reshape(arr.shape) bytestr = bytestr[arr.nbytes:]
```

Fig. 1. Representing the model as bytes in Python

In the implemented design, the global model, as well as the mean and standard deviation data, are stored as properties within the smart contract. Conversely, the local counterparts of these values are managed using events in Solidity, as illustrated in Figure 2. This design choice is well-suited for the specific use case, as events offer a more appropriate mechanism for handling and tracking local updates. Additionally, this decision helps to minimize the gas fees incurred by the clients, as events are less costly compared to direct on-chain storage. Alternatively, an option exists to allow clients to transmit their local updates to the server outside the blockchain. Some clients may prefer this approach, particularly due to the limited privacy protections offered by Ethereum or the most open blockchains in general [20]. In the proposed setup, this hybrid configuration is also feasible, where some clients communicate through the blockchain while others utilize external channels. This flexibility accommodates the privacy preferences of clients and acknowledges the constraints imposed by the blockchain system.

One advantage of an all-blockchain approach is that it allows for third-party verification of the server's correct execution of the model averaging process. By conducting all interactions within the blockchain network, external entities can scrutinize and validate the integrity of the model averaging performed by the server. This transparency enhances the trust and accountability associated with the FL system, fostering confidence among participants.

Ensuring that only the server (specifically, the contract owner in Solidity) performs the global updates is of paramount importance to maintaining the integrity and security of the system. To address this concern, a modifier named "Owner Only" is implemented. This modifier serves as a safeguard, allowing only the contract owner to execute the global updates. By restricting the update functionality to the authorized server, the risk of unauthorized modifications or tampering with the global model is mitigated.

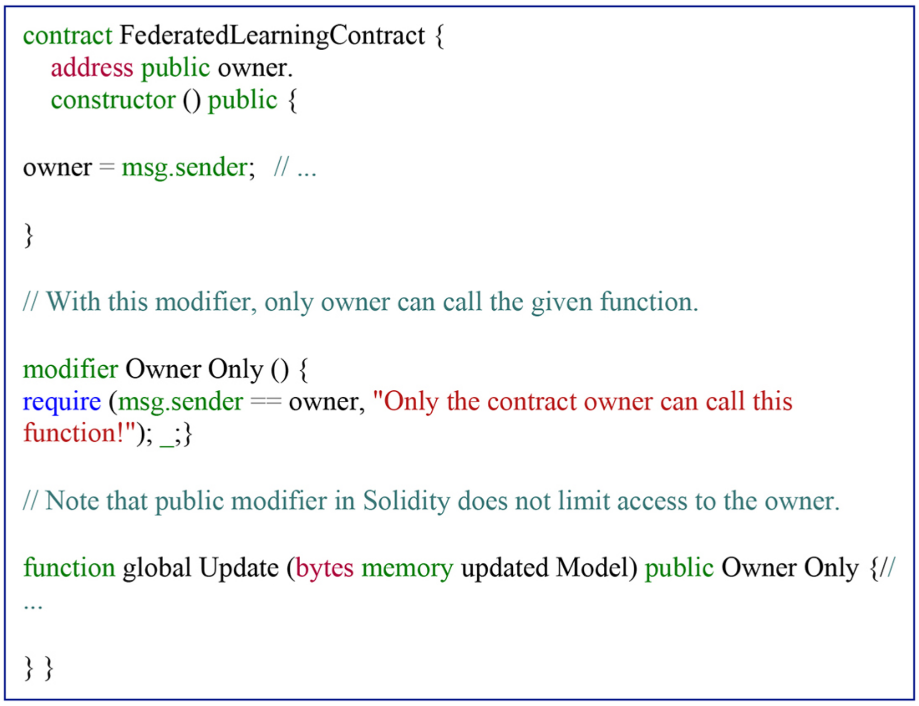
The combination of utilizing events for local updates, providing flexibility for clients to communicate inside or outside the blockchain, and enforcing the "Owner Only" modifier contributes to the robustness, privacy, and security of the blockchain-based FL system. These design decisions address key considerations in the implementation, promoting the effectiveness and integrity of the training process while accommodating privacy preferences and facilitating verifiability within the decentralized network.

```
// Representation of the global model in bytes.

bytes public model;
// For data standardization, global means and stds. bytes public means;
bytes public stds;

// This event is fired when a client reports a local update.

event Local Update (address indexed from, unit epoch, unit size, bytes
model).

// These events are fired during the preprocessing stage
// by the clients.
event Local Means (address indexed from, unit size, bytes data); event Local
Stds(address indexed from, unit size, bytes data).
```

Fig. 2. FL data in solidity

Relevant parts of the code are given in Figure 2. This might seem to create a single point of failure, but since the transactions are in the blockchain, anyone can take the latest (or a previous) version of the model and create a new contract. At first glance, this approach may appear to introduce a single point of failure, as the model and its associated data are stored within a single smart contract. However, due to the transparent nature of blockchain transactions, any participant can access the latest version of the model or even a previous version and create a new contract based on that information. The decentralized and immutable nature of the blockchain ensures that all transactions, including updates to the model, are recorded and publicly available. This transparency empowers any participant within the network to retrieve the latest model state and initiate the creation of a new contract based on that information. By leveraging the blockchain's inherent transparency and immutability, the risk of a single point of failure is mitigated. In the event of a failure or loss of access to the existing contract, participants can rely on the recorded transactions and the availability of the model's latest version to recreate the system. This ability to recreate the contract based on historical data allows for continuity and resilience in the face of potential failures or other adverse events.

Furthermore, the openness of the blockchain network enables a diverse set of participants to validate and verify the model's integrity. By examining the transactions and data stored on the blockchain, any interested party can independently recreate the model and assess its accuracy and reliability. This decentralization of verification helps to enhance trust and mitigate the risks associated with a single point of failure.



```
contract FederatedLearningContract {
   address public owner.
   constructor () public {

owner = msg.sender;   // ...

}

// With this modifier, only owner can call the given function.

modifier Owner Only () {
require (msg.sender == owner, "Only the contract owner can call this
function!"); _;}

// Note that public modifier in Solidity does not limit access to the owner.

function global Update (bytes memory updated Model) public Owner Only {//
...

} }
```

Fig. 3. Owner only modifier in solidity

Enums are employed to maintain a record of the training stage, as depicted in Figure 4. The server assumes the role of orchestrating the training process by transmitting global mean and standard deviation data and updates to the clients.

To ensure clarity and simplicity, the implementation of the learning system on the blockchain focused solely on the core FL functionality and disregarded other potential transactions that may occur between the clients and the server. Nevertheless, incorporating additional features into the system is relatively straightforward, given the foundational framework of blockchain-based federated learning. For instance, it is conceivable to introduce mechanisms where the server compensates the clients for their local updates and the reporting of local μk and σk values. This compensation mechanism serves the dual purpose of addressing privacy concerns and providing a financial incentive for clients to actively participate in the training process. By enabling such supplementary features, the system can address the privacy concerns associated with federated learning, as clients may be hesitant to share sensitive data without adequate safeguards. Compensating clients for their contributions can help alleviate these concerns by creating a sense of trust and incentivizing their continued engagement in the training process. Moreover, introducing a financial incentive promotes active participation from clients, potentially enhancing the overall effectiveness and efficiency of the FL system.

While this study primarily focuses on the foundational aspects of blockchain-based federated learning, the extensibility of the system allows for the inclusion of additional features to address practical considerations and incentivize participation. Future research and development can explore the integration of financial incentives and privacy-enhancing mechanisms to further optimize the blockchain-based FL system and promote its adoption in real-world scenarios.

```
contract FederatedLearningContract {
// Enum representing current stage of the federated learning model Enum
Stage {PREPROCESS_MEANS, PREPROCESS_STDS, TRAINING}
Stage public stage.

constructor () public {// ...

stage = Stage.PREPROCESS_MEANS;}

function local Means (unit size, bytes memory data) public {
require (stage == Stage.PREPROCESS_MEANS, "Can only be called in
means □→ preprocessing stage!");
emit Local Means (msg.sender, size, data); }

function global Means (bytes memory data) public Owner Only {
require (stage == Stage.PREPROCESS_MEANS, "Can only be called in
means □→ preprocessing stage!");
means = data;
// Advance stage
stage = Stage.PREPROCESS_STDS;}

   // Similar for other functions   }
```

Fig. 4. Training stages in sol

3.4 Federated learning on blockchain

The utilization of FL on the blockchain introduces specific challenges that require a customized approach to address them effectively. One of the primary challenges pertains to the gas fees associated with storing byte-sized data on the blockchain. Gas fees serve as incentives for maintaining efficient use of blockchain resources, but they also impose limitations on the size of models that can be practically constructed and trained on the blockchain network.

To tackle the challenge of gas fees, an approach was implemented in the FL system to modify the precision level of the model stored on the blockchain. This modification enables clients to employ high-precision double floating-point numbers during local updates while transmitting the results to the blockchain in compressed formats such as 32-bit or even 16-bit floating-point representations. By doing so, the gas fees associated with training the model on the blockchain network are significantly reduced. This precision modification option allows for more efficient use of blockchain resources while maintaining an acceptable level of accuracy for the FL model.

In addition to the financial incentives related to smaller model sizes on the blockchain network, there are already several other reasons in the field of ML to keep model sizes small. These reasons include preventing overfitting, which occurs when a model becomes too complex and fits the training data too closely, leading to poor generalization on new, and unseen data. Additionally, smaller model sizes facilitate faster training times, allowing for more efficient use of computational resources.

Consequently, the integration of FL on the blockchain necessitates a more nuanced and careful approach to model development and training. It requires striking a balance between the size of the model, its accuracy, and the associated gas fees. This calls for a thoughtful consideration of various factors, such as the specific requirements of the FL application, the available computational resources, and the constraints imposed by the blockchain network.

Last but not least, leveraging FL on the blockchain poses unique challenges due to gas fees associated with storing data. To overcome these challenges, the precision level of the model can be modified, allowing for reduced gas fees while maintaining an acceptable level of accuracy. The advantages of smaller model sizes in machine learning, such as preventing overfitting and achieving faster training times, further reinforce the need for a careful and nuanced approach when incorporating FL on the blockchain.

4 RESULTS

4.1 Blockchain limitations

To assess the limitations of the blockchain system, a testing script was developed to measure the gas fees associated with various model sizes. The script evaluated the gas fee required to perform one global and ten local updates, which equates to approximately one round of updates, on a test blockchain setup. Table 1 presents the gas fees for different byte sizes and their corresponding ratios to the gas fee for one byte. It is important to note that the gas fees impose a significant constraint on the maximum model size that can be stored on the blockchain. The table also includes the ratio of each gas fee to the base gas fee, which is set at a value of 736,795 Gwei for a byte size of 1. The following is a breakdown of the table:

- Byte Size: This column represents the size of the data in bytes.
- Gas Fee (Gwei): It denotes the associated gas fee, measured in Gwei, for each specific byte size. Gas fees are typically used in blockchain networks to compensate miners or validators for executing transactions or computations.
- Ratio to Base Gas Fee: This column illustrates the ratio of each gas fee to the base gas fee of 736,795 Gwei for a byte size of 1. For instance, a ratio of 1.0016 means that the gas fee for a byte size of 10 is 0.16% higher than the base gas fee.

The table reveals that as the byte size increases, the gas fee tends to rise. For example, a byte size of 10 incurs a slightly higher gas fee compared to a byte size of 1, as indicated by the ratio of 1.0016. As the byte size further increases, the gas fee escalates significantly. At a byte size of 100, the gas fee is 1.1426 times the base fee, and at a byte size of 1,000, it rises to 2.2110 times the base fee.

The trend continues, with larger byte sizes resulting in considerably higher gas fees. At a byte size of 10,000, the gas fee jumps to 14.1934 times the base fee, and at a byte size of 20,000, it surges to 27.5460 times the base fee. This pattern persists until a byte size of 50,000, at which point the system runs out of gas, as indicated in the table. Overall, the table showcases the relationship between byte size and gas fees, illustrating how the gas fee increases proportionally to the size of the data being processed, up to a certain limit where the system becomes unable to handle the transaction due to insufficient gas.

Table 1. Byte size vs. gas fee

Byte Size	Gas Fee (Gwei)	Ratio to Base Gas Fee
1	736795	1.0000
10	737983	1.0016
100	841886	1.1426
1000	1629055	2.2110
10000	10457631	14.1934
20000	20295764	27.5460
30000	30004373	40.7228
40000	39810720	54.0323
50000	Out of gas	Out of gas

In the test system, all accounts were initially endowed with a substantial amount of 1,000,000 ETH, which is approximately equivalent to 1.2 billion dollars at the time of writing. The experimental outcomes revealed that once the model size reached 50,000 bytes, it became impractical to execute even a single round of updates on the blockchain with the given amount of ETH. These results highlight the critical importance of considering the limitations imposed by the blockchain system when designing and implementing FL models.

To ensure the financial viability of the model, it becomes necessary to carefully manage its size and associated costs. For instance, consider a model comprising 100 inputs (features), a single hidden layer with 50 neurons, and 10 outputs (classes). This model would necessitate $100 \times 50 + 50 \times 10 = 5{,}500$ weights and $50 + 10 = 60$ biases, resulting in a total of 5,560 floating-point numbers. If 16, 32, or 64-bit floats are utilized, the byte size of the model would be 11,120, 22,240, and 44,480, respectively. It is worth noting that gas fees for communicating means and standard deviations tend to be significantly lower since they involve one-dimensional data with a length equal to the number of features. This reduction in complexity compared to the model itself helps mitigate the gas fees associated with transmitting these statistical values in the blockchain network.

Considering the substantial costs associated with gas fees, it becomes imperative to optimize the model size and communication strategies within the FL framework. This optimization can involve techniques such as parameter compression, quantization, or other model size reduction methods to reduce the overall byte size without compromising the model's performance. By carefully managing the size of the model and its associated communication overhead, researchers and practitioners can ensure the financial viability and practicality of FL models within blockchain systems. For instance, in our scenario, the gas fees for communicating means and standard deviations tend to be a lot less since it's just 1D data with its length equal to the number of features.

4.2 Experiments and findings

In this study, the dataset provided in reference [11], consisting of 25,192 labeled samples, was employed for the experiments. The dataset encompasses 41 features, including 3 categorical variables, and comprises two classes: anomaly and normal.

To establish the training and validation datasets, an 80%/20% split was applied, resulting in approximately 20,000 samples allocated for training. These training samples were randomly assigned to 10 clients, adopting a learning rate of 0.01, unless specified otherwise.

Initially, a single hidden layer with 50 neurons and a rectified linear unit (ReLU) activation function was employed. The validation set achieved a maximum accuracy of 99.17% under the settings of C = Cpre = 1, 10 global epochs, 5 local epochs, and a batch size of 32.

During the course of the experiments, it was observed that modifying the precision level of the model on the blockchain, whether using 16-bit, 32-bit, or 64-bit floating-point numbers, did not yield a significant impact on the results. This alteration solely influenced how the model was communicated, not its internal representation, which consistently utilized 64-bit floating-point numbers. Specifically, clients obtained the global model from the blockchain in 16-bit floating-point format, converted these values to 64-bit floats, performed gradient descent, and reported their updates back to the blockchain using 16-bit floating-point numbers.

Furthermore, it was noted that the configuration of the hidden layers had minimal influence on the final results. Through progressively simplifying the model, a single linear layer with an output of 2 neurons, represented as Ax + B, was employed. Despite the simplicity of this model, it achieved an accuracy of 97.28%. The resulting byte size was 952 when using 32-bit floats, or 476 when using 16-bit floats.

These findings underscore the resilience of the FL approach in the face of precision changes and model simplification. They demonstrate that the accuracy of the model can be maintained while reducing the byte size on the blockchain. Such observations provide insights into the potential optimization of model representation and communication in blockchain-based FL systems. Lastly, the experimental results highlight the robustness of the implemented approach in the intrusion detection domain. The findings contribute to understanding the impact of different precision levels and model architectures, suggesting avenues for further investigation and optimization in blockchain-based FL scenarios.

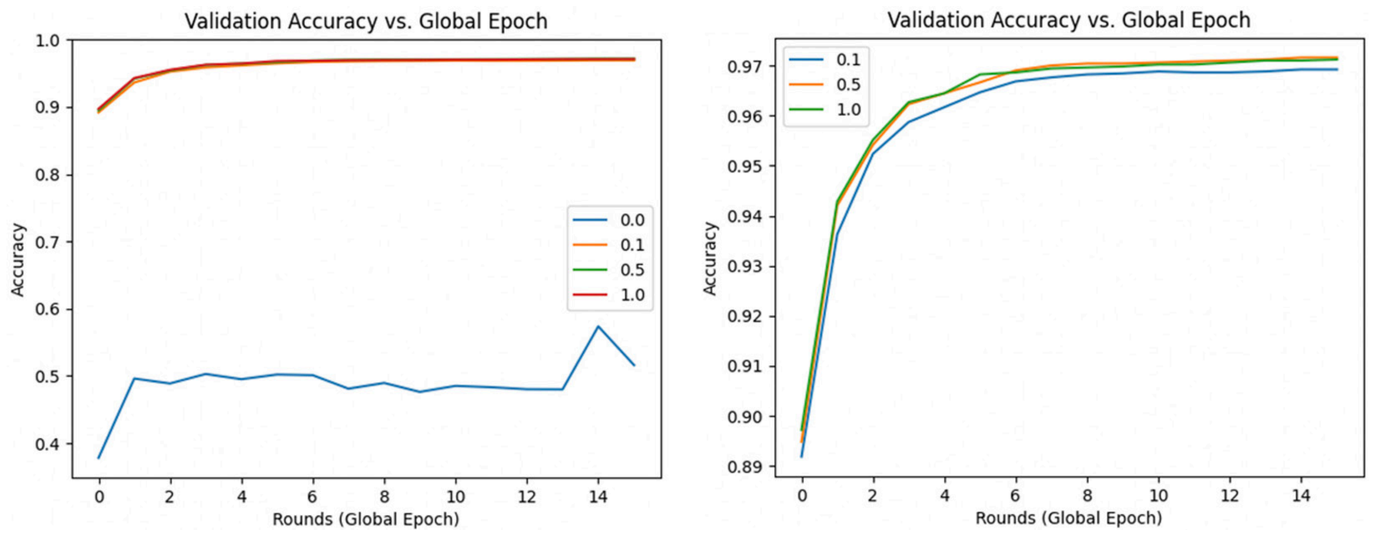


Fig. 5. Accuracy vs. epoch plots with varying Cpre

In this study, it was found that data standardization was crucial for achieving high accuracy on the given dataset, as the accuracy significantly dropped to

50% without it. Various hyperparameters and learning rates were tested, but none were successful in improving accuracy without data standardization. Moreover, the use of 16-bit floats resulted in NaN values without standardization. To further investigate the effects of data standardization, the learning rate was decreased to 0.0001, and the number of global epochs was increased to 16 while experimenting with different Cpre values while keeping other parameters constant. The results showed that even when only one client shared the mean and standard deviation data, the accuracy was nearly as high as the case where all clients shared the data, indicating the significance of data standardization. However, this may not hold true for non-IID data, which is typical in real-world scenarios. The Cpre value was then fixed to 0.5, and different C values were tested, but the differences were negligible. Finally, the model was evaluated on 10% of the larger and more complex KDD Cup 1999 dataset with 23 class labels. Despite the increased size and complexity of the dataset, the single-layer linear model still achieved 98% accuracy on the validation set, as long as the data standardization step was performed.

5 CONCLUSION

This study encompasses an exploration of Ethereum, Solidity, and blockchain technologies, along with the incorporation of FL techniques. While our contribution to FL is limited, we have identified a unique aspect of the data standardization stage that is absent from existing literature. Furthermore, we have addressed challenges arising from applying FL within a blockchain system, specifically model size restrictions. By adjusting a single configuration parameter, we achieved a halving of model size on the blockchain without significant accuracy loss. Although initially focused on intrusion detection, our project can be readily adapted for diverse blockchain-based FL applications. Moreover, our modular platform structure allows for the development of alternative blockchain backends. In conclusion, this research expands our understanding of Ethereum, Solidity, and blockchain while advancing FL methodologies. Future investigations in this domain hold potential for enhancing privacy, scalability, and efficiency in decentralized ML systems.

6 REFERENCES

[1] Z. Wang and Q. Hu, “Blockchain-based federated learning: A comprehensive survey,” *arXiv preprint arXiv: 2110.02182*, 2021. https://doi.org/10.48550/arXiv.2110.02182

[2] J. Alzubi, A. Nayyar, and A. Kumar, “Machine learning from theory to algorithms: An overview,” *Journal of Physics: Conference Series*, vol. 1142, p. 012012, 2018. https://doi.org/10.1088/1742-6596/1142/1/012012

[3] H. Yan, Y. Liu, Z. Zhang, and Q. Wang, “Efficient privacy-preserving certificateless public auditing of data in cloud storage,” *Security and Communication Networks*, vol. 2021, no. 1, pp. 1–11, 2021. https://doi.org/10.1155/2021/6639634

[4] A. Vennala, M. Radha, M. Rohini, M. Anees Fathima, and P. D. Lakshmi, “Efficient privacy-preserving certificateless public auditing of data in cloud storage,” *J. Eng. Sci.*, vol. 13, pp. 532–541, 2022. https://www.hindawi.com/journals/scn/2021/6639634/

[5] N. A. Akbar, A. Muneer, N. ElHakim, and S. M. Fati, “Distributed hybrid double-spending attack prevention mechanism for proof-of-work and proof-of-stake blockchain consensuses,” *Future Internet*, vol. 13, no. 11, p. 285, 2021. https://doi.org/10.3390/fi13110285

[6] Y.-N. Liu, Y.-P. Wang, X.-F. Wang, Z. Xia, and J.-F. Xu, “Privacy-preserving raw data collection without a trusted authority for IoT,” *Computer Networks*, vol. 148, pp. 340–348, 2019. https://doi.org/10.1016/j.comnet.2018.11.028

[7] M. Kamran and A. A. A. Almaghthawi, “Case-based reasoning diagnostic system for antenatal research database,” *International Journal of Online & Biomedical Engineering*, vol. 18, no. 7, pp. 176–187, 2022. https://doi.org/10.3991/ijoe.v18i07.30353

[8] D. C. Nguyen, M. Ding, P. N. Pathirana, A. Seneviratne, and A. Y. Zomaya, “Federated learning for COVID-19 detection with generative adversarial networks in edge cloud computing,” *IEEE Internet of Things Journal*, vol. 9, no. 12, pp. 10257–10271, 2022. https://doi.org/10.1109/JIOT.2021.3120998

[9] D. Li, Z. Luo, and B. Cao, “Blockchain-based federated learning methodologies in smart environments,” *Cluster Computing*, vol. 25, no. 4, pp. 2585–2599, 2022. https://doi.org/10.1007/s10586-021-03424-y

[10] L. Li, Y. Fan, M. Tse, and K.-Y. Lin, “A review of applications in federated learning,” *Computers & Industrial Engineering*, vol. 149, p. 106854, 2020. https://doi.org/10.1016/j.cie.2020.106854

[11] A. Nilsson, S. Smith, G. Ulm, E. Gustavsson, and M. Jirstrand, “A performance evaluation of federated learning algorithms,” in *Proceedings of the Second Workshop on Distributed Infrastructures for Deep Learning*, 2018, pp. 1–8. https://doi.org/10.1145/3286490.3286559

[12] S. Agrawal *et al.*, “Federated learning for intrusion detection system: Concepts, challenges and future directions,” *Computer Communications*, vol. 195, pp. 346–361, 2022. https://doi.org/10.1016/j.comcom.2022.09.012

[13] S. K. Wagh, V. K. Pachghare, and S. R. Kolhe, “Survey on intrusion detection system using machine learning techniques,” *International Journal of Computer Applications*, vol. 78, no. 16, pp. 30–37, 2013. https://doi.org/10.5120/13608-1412

[14] W. Li, W. Meng, and M. H. Au, “Enhancing collaborative intrusion detection via disagreement-based semi-supervised learning in IoT environments,” *Journal of Network and Computer Applications*, vol. 161, p. 102631, 2020. https://doi.org/10.1016/j.jnca.2020.102631

[15] B. Cetin, A. Lazar, J. Kim, A. Sim, and K. Wu, “Federated wireless network intrusion detection,” in *2019 IEEE International Conference on Big Data (Big Data)*, 2019, pp. 6004–6006. https://doi.org/10.1109/BigData47090.2019.9005507

[16] O. Alkadi, N. Moustafa, B. Turnbull, and K.-K. R. Choo, “A deep blockchain framework-enabled collaborative intrusion detection for protecting IoT and cloud networks,” *IEEE Internet of Things Journal*, vol. 8, no. 12, pp. 9463–9472, 2020. https://doi.org/10.1109/JIOT.2020.2996590

[17] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, “Communication-efficient learning of deep networks from decentralized data,” in *Artificial Intelligence and Statistics*, 2017, pp. 1273–1282. http://proceedings.mlr.press/v54/mcmahan17a?ref=https://githubhelp.com

[18] A. Muneer, R. F. Ali, A. Alghamdi, S. M. Taib, A. Almaghthawi, and E. A. A. Ghaleb, “Predicting customers churning in banking industry: A machine learning approach,” *Indones. J. Electr. Eng. Comput. Sci.*, vol. 26, no. 1, p. 539, 2022. https://doi.org/10.11591/ijeecs.v26.i1.pp539-549

[19] H. Anysz, A. Zbiciak, and N. Ibadov, “The influence of input data standardization method on prediction accuracy of artificial neural networks,” *Procedia Engineering*, vol. 153, pp. 66–70, 2016. https://doi.org/10.1016/j.proeng.2016.08.081

[20] S. Tikhomirov, “Ethereum: State of knowledge and research perspectives,” in *Foundations and Practice of Security, (FPS 2017),* in Lecture Notes in Computer Science, A. Imine, J. Fernandez, J. Y. Marion, L. Logrippo, and J. Garcia-Alfaro, Eds., Springer, Cham., vol. 10723, 2018, pp. 206–221. https://link.springer.com/chapter/10.1007/978-3-319-75650-9_14

7 AUTHORS

Ahmed Almaghthawi is with the Department of Computer Science, College of Science and Art at Mahayil, King Khalid University, Abha 62529, Saudi Arabia (E-mail: aalmaghthwi@kku.edu.sa).

Ebrahim A. A. Ghaleb is with the Department of Computer and Information Sciences, University Teknologi PETRONAS, Seri Iskandar 32160, Malaysia.

Nur Arifin Akbar is with the Department of Computer and Information Sciences, University Teknologi PETRONAS, Seri Iskandar 32160, Malaysia.

Layla Asiri is with the Department of Computer Information System, Applied College at Mahayil, King Khalid University, Abha 62529, Saudi Arabia.

Meaad Alrehaili is with the Department of Computer Sciences and Artificial Intelligence, Collège of Computer Science, and Engineering, University of Jeddah, Jeddah, Saudi Arabia.

Askar Altalidi is with the Department of Information Science, College of Science & Art at Mahayil, King Khalid University, Abha 62529, Saudi Arabia.