MDPI

Article

Blockchain-Enabled: Multi-Layered Security Federated Learning Platform for Preserving Data Privacy

Zeba Mahmood [1,*] and Vacius Jusas [2]

1   Department of Software, Kaunas Technology University, 44249 Kaunas, Lithuania
2   Faculty of Informatics, Kaunas Technology University, 44249 Kaunas, Lithuania; vacius.jusas@ktu.lt
*   Correspondence: zeba.mahmood@ktu.edu

Abstract: Privacy and data security have become the new hot topic for regulators in recent years. As a result, Federated Learning (FL) (also called collaborative learning) has emerged as a new training paradigm that allows multiple, geographically distributed nodes to learn a Deep Learning (DL) model together without sharing their data. Blockchain is becoming a new trend as data protection and privacy are concerns in many sectors. Technology is leading the world and transforming into a global village where everything is accessible and transparent. We have presented a blockchain enabled security model using FL that can generate an enhanced DL model without sharing data and improve privacy through higher security and access rights to data. However, existing FL approaches also have unique security vulnerabilities that malicious actors can exploit and compromise the trained model. The FL method is compared to the other known approaches. Users are more likely to choose the latter option, i.e., providing local but private data to the server and using ML apps, performing ML operations on the devices without benefiting from other users' data, and preventing direct access to raw data and local training of ML models. FL protects data privacy and reduces data transfer overhead by storing raw data on devices and combining locally computed model updates. We have investigated the feasibility of data and model poisoning attacks under a blockchain-enabled FL system built alongside the Ethereum network and the traditional FL system (without blockchain). This work fills a knowledge gap by proposing a transparent incentive mechanism that can encourage good behavior among participating decentralized nodes and avoid common problems and provides knowledge for the FL security literature by investigating current FL systems.

Keywords: Federated Learning (FL); blockchain; Zero-Knowledge Proofs (ZKPs); multi-layer security; poisoning attacks; Convolutional Neural Network (CNN)

1. Introduction

Over the past decade, mobile devices and internet use have increased exponentially [1,2]. Given the exponential use of mobile devices, many groups have begun to investigate how these devices Could be used effectively to collect data. The availability of powerful sensors such as microphones, cameras, and global positioning systems (GPS) on these devices, and the fact that they are frequently carried and used, means that they have access to unprecedented amounts of data. DL Models learned from such data can greatly improve the user experience by enabling smarter applications [3]. Training data resides on a single, centralized server in a traditional DL approach. Consequently, DL companies must go through a lengthy and costly process of evaluating their users' data, regardless of the risks and responsibilities of storing such data sets in a centralized data center. DL Model becomes more complex as such data grows in size and prototype, disenfranchising smaller start-ups and monopolizing the market. Moreover, data privacy remains a major challenge for established companies. In 2019, for example, Facebook and Amazon admitted to wiretapping users' recordings without their consent. Rising online protests over how companies handle users' private data have forced regulators to step in and force these

organizations to stop the practice [4]. FL addresses these issues by allowing geographically distributed nodes (also called clients) to learn a shared global model without sharing their training data [5]. FL allows personal data to remain on multiple devices in local locations, minimizing the possibility of data breaches. FL is essentially a decentralized DL approach that decouples training data from a globally shared model on personal devices. Recently, FL has become a leading research area in the field of artificial intelligence (AI) and has privacy and security mechanisms that can comply with current data protection laws such as the General Data Protection Regulations (GDPR) [6]. Despite its potential, FL is not immune to some forms of attacks that can target any stage of the network's training and deployment pipelines. Common attacks that can compromise FL systems include backdoor attacks [7], poisoning attacks [8], and inference attacks [9]. By sharing model parameters instead of training data, FL creates new attack surfaces those malicious actors can exploit and compromise the system. Thus, an attacker who has full control over one of the participating nodes can maliciously manipulate the training data, training pipelines, and model parameters. The decentralized nature of FL, especially when augmented with secure aggregation, can lead to many attacks going unnoticed in FL [10]. Secure aggregation prevents the central server from checking the model update of each node. Moreover, when training the DL model with thousands of distributed mobile devices, it is impossible to detect whether any of them is malicious or compromised. The security issues faced by FL's current systems are trust issues that blockchain can solve.

As a secure technology, Blockchain can withstand a single point of failure because it relies on distributed consensus. In addition to improving security, consensus protocols can help build trust in the learning process [11]. FL was first defined by researchers in 2016 and has since gained global traction, both in academia and industry. FL was developed to shift the training effort to the device itself, federating local models and learning processes and the devices with ML. Its main goal is to provide a framework for privacy-preserving machine learning. The FL method is compared with the other known approaches on the right-hand side of Figure 1. Users are more likely to choose the latter option between providing local but private data to the server and using ML apps, performing ML operations on devices without benefiting from other users' data, preventing direct access to raw data, and incorporating locally training ML models. FL protects data privacy and reduces data transfer overhead by storing raw data on the devices and combining locally computed model updates. Although there are reviews of both edge computing (EC) and federated learning (FL), most studies treat the two areas separately. Moreover, most writings have not considered the difficulties in device requirements. In [12], the structures and frameworks for edge insights were considered. They provided insights that are explicitly for 6G organizations. Organizations whose data sets are getting bigger and bigger need to communicate faster and deliver their products quickly over the Internet. Therefore, such organizations need a faster and more credible network.

On the other hand, ref. [13] investigated the use of ML for the IoT board. It also investigated the open doors and difficulties in federated learning. The authors of [14] presented the current federated learning and proposed the design of federated learning frameworks. Another focus was to present the design and characterization of different federated authors, proposed a scientific categorization of federated learning and divide the federated learning framework into six different perspectives: Information dissemination, ML model, protection tool, correspondence design, alliance size and inspiration for federated learning. The authors have presented the factors of the plan, the contextual analyses and the open doors for future research. The essence of this is that they have given a brief guide to FL and the difficulties of federated learning, provided a comprehensive overview of the writing, and pointed out some future research directions. They have reflected on four difficulties of federated learning: productivity of correspondence, heterogeneity of setting, substantial heterogeneity, and safety. Learning arrangements [15] for different information transmissions utilized for preparing and are even utilized in planning blockchain-based administrations and securities [16]. The stochastic game executes stochastic advancements

under the conditions of the game, and the players in a stochastic game might change their methods considering the past activities and irregularity of the behavior of different players. Some types of stochastic games have been adapted to avoid the 51% attack, and a stochastic game has been used to study the decision between fair mining and choosing the appropriate time to add and deliver mining blocks. The blockchain can also encourage participants to contribute positively to the system FL via an incentive mechanism that rewards good behavior. A blockchain-powered FL system can overcome challenges such as poisoning attacks, inference attacks and backdoor attacks by strengthening trust and providing incentive mechanisms that reward good behavior and punish bad actors. To take advantage of FL and exploit its features, it is necessary to know potential security attacks in these systems and how to avoid them. Our paper proposes a blockchain-enabled FL with a multi-layered security model. The rest of the paper is divided into five sections as follows: Section 2 explores the various vulnerabilities and attacks inherent in FL systems; Section 3 examines the findings on existing defense mechanisms and their limitations, and Section 4 extends the defense mechanisms to blockchain and other peer-to-peer (P2P) networks. Finally, Section 5 concludes.
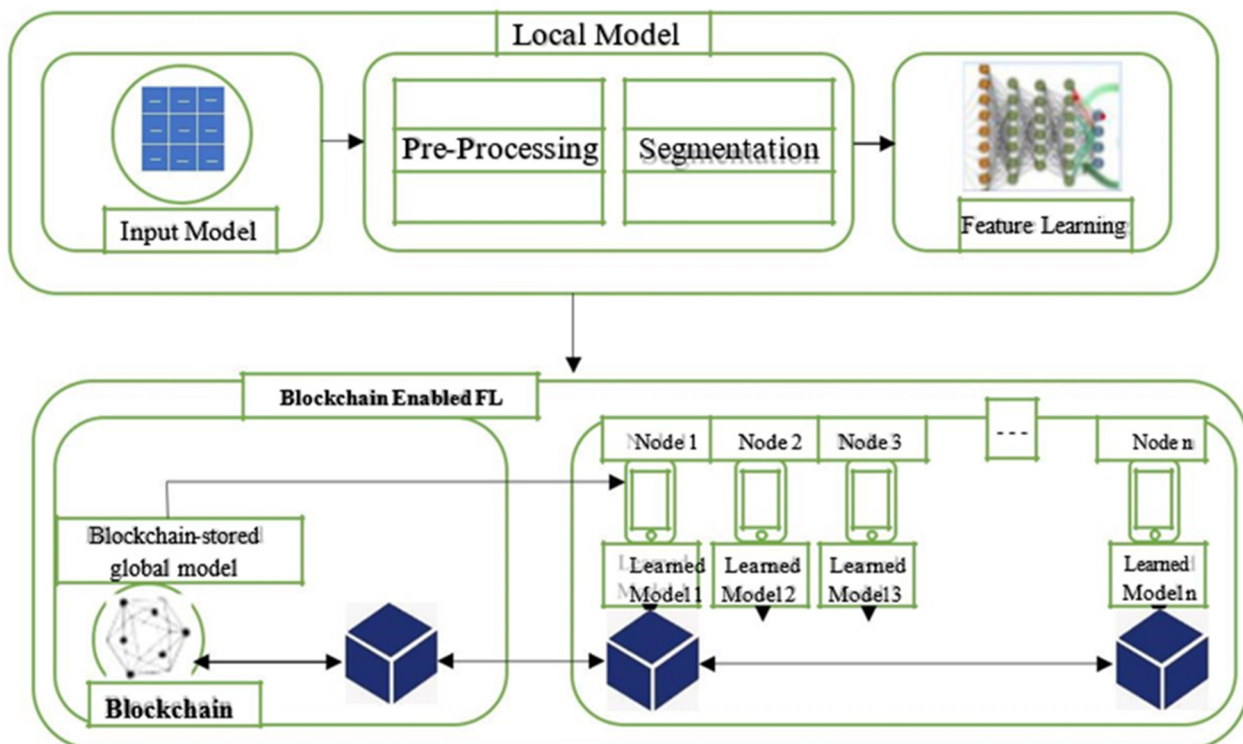

Figure 1. System Description.

1.1. Statement and Purpose

FL has evolved to implement a distributed stochastic optimization for deep learning framework training [11]. A typical FL strategy involves multi-round training where each round has the following steps:

- Selecting nodes. The centralized server samples the nodes from a set of participating clients that meets requirements such as Wi-Fi connection, device idle, device plugged in etc.
- Sharing the global model. Each selected node downloads the global model from the central server.
- Local training. Each node computes an update to the DL model based on its training data and the training procedure, such as local stochastic gradient descent (SGD).

- Aggregation. Each node transmits its output to the server. The server, in turn, aggregates the updates to generate an improved global model.
- Updating the global model. The global model gets updated with the new aggregate emanating from participating nodes.

The FL architectural framework introduces unique security vulnerabilities from three sources: nodes, aggregators (server), and third parties. A malicious node (containing malware) that gains complete control over one or multiple nodes can compromise the entire FL model in various forms such as:

- Manipulating the training data of any compromised endpoint.
- Tampering or replacing the resulting local model's parameters before transmitting them to the central server.
- Manipulating the training procedure, including local loss optimisation and other parameters such as local learning rate, batch size, and local epochs.
- Dynamically modifying the training procedure from one round to another.

Similarly, a hostile aggregator can launch targeted and untargeted attacks on the shared global model.

Aggregator integrity certification methods such as zero-Knowledge Proofs (ZKPs) [17] and Secure MultiParty Computations (SCMs) [18] can detect these attacks. However, reasonably honest aggregators can still infer private information from model updates.

Finally, third parties can also compromise the security of a federated learning system. This can be the case when nodes exchange model updates during the training phase, potentially exposing the model to external attackers [18]. In this context, an attacker who eavesdrops on the communication channel can effectively derive the private data of the nodes and compromise the entire FL system.

1.2. Literature Review

1.2.1. Security Attacks in Federated Learning Systems

There are two forms of attacks in typical FL systems, distinguished by their goals: targeted and untargeted attacks. In targeted attacks (also called backdoor attacks), the attacker alters the model's behavior by focusing on specific subtasks within the model and maintaining the overall accuracy of the main task. For example, an attacker may corrupt an image classification system and misclassify "green cars" as "frogs" while ensuring that other cars are correctly classified. In an untargeted attack, the malicious actor attempts to degrade the overall accuracy of the global model. The following subsection classifies security attacks in FL systems and briefly describes each attack.

1.2.2. Poisoning Attacks

Poisoning attacks are among the most common security attacks in FL systems [19]. In poisoning attacks, an attacker who has gained control over the participating nodes can poison the training data by injecting carefully designed samples to compromise the entire federated learning experience. This is because each node has access to the training data and the likelihood of manipulating such data weights is very high. These attacks target various forms of FL artifacts, including Data poisoning. With data poisoning attacks, adversaries generate "Not Accurate" samples on the compromised nodes to generate falsified parameters and transmit them to the central server. Model poisoning. Unlike data poisoning, where the attacker manipulates the global model via fake training data, model poisoning involves poisoning the global model itself. Then, the adversary modifies the updated model before transmitting it to the central server for aggregation.

1.2.3. Backdoor Attacks

In a backdoor attack, a malicious actor inserts a hidden malware into the global model while preserving the accuracy of the primary task [20]. In this regard, compromised nodes train the model on the chosen backdoor data that poisons the entire FL pipeline. These attacks can harm FL systems as they can predict false results with high confidence levels. Furthermore, a shared global model that averages across participating nodes is shared in an FL-based system [21]. Therefore, an attacker who knows the global model state can easily apply a simple weight re-scaling operation and replace the model. For backdoor attacks to be successful, they must satisfy two requirements: the model must be close to convergence, and the adversary must have near-perfect knowledge of some FL parameters such as the number of nodes and data size [16].

1.2.4. Inference Attacks

A serious gradient leakage can occur when nodes exchange gradients during the FL training process. In this case, the model updates can leak information regarding the nodes' training data features to malicious actors. While some of these features are not necessarily related to the main task, they impact gradient leakage. A malicious actor can also take a snapshot of the model parameters and perform property inferences by computing the difference between the consecutive snapshots. This is like aggregating model updates from all participating nodes minus the malicious actor. In this case, the adversarial node generates gradients based on the node's private data. Most deep neural networks (DNNs) compute the gradient of a particular layer using its features and backpropagate error from the following layer. For example, the gradients in sequential, fully connected layers are simply the products of activations and errors backpropagated from the successive layers. The exact process occurs in the convolutional layer, where gradients are simply convolutions of the errors from the layer above and its activations. An adversary with access to this model update can obtain a substantial volume of private data, including class representations [22] class memberships [4] and properties of other nodes' data. An adversary can infer labels from the gradients in a worst-case scenario and retrieve the original training data without prior knowledge about the training set [8].

1.2.5. Machine Learning Approaches

Regression methods are used for training supervised ML. The goal of regression techniques is usually to explain or predict a numerical value using a previous data set. For example, regression methods may use historical price data and then predict the price of a similar property to forecast retail demand. Linear regression is considered the simplest and most basic method. In this case, a data set is modelled using the following equation:

$$(y = m \times x + b) \tag{1}$$

It is possible to train a regression model with several pairs of data, e.g., x, y. To do this, one must define a position and the slope of the line with a minimum distance to all known data points. This line best approximates the data observations and can help make predictions for new data not yet seen.

1.2.6. Clustering

Unsupervised learning methods are clustering algorithms. K-means, mean-shift and expectation maximization are three typical clustering techniques. In commercial applications, grouping or clustering techniques are helpful when segmenting or categorizing large amounts of data. For example, customers can be segmented based on different characteristics to better target marketing campaigns, and messages that appeal to confident readers can be recommended. Clustering is particularly useful for identifying patterns in large data sets that are not visible to the naked eye. Therefore, one of the most commonly used AI approaches in marketing.

1.2.7. Existing Defences against Security Attacks in FL Systems and Their Limitations

While security and privacy preservation have extensively been studied in the machine learning community, it has yet to receive attention in federated learning. FL environments have statistical heterogeneity in data sets and sporadic access to network connectivity and power. Existing works on defences against security in FL systems largely focus on anomaly detection [23] differential privacy [24], Secure aggregation [25], Pruning [26], ZKPs [27], Adversarial training [28] and federated distillation (FD) [29].

1.2.8. Anomaly Detection

Anomaly detection can help discover and thwart attacks such as model poisoning and data poisoning in FL systems via various methods. One popular method is using the test error rate [30]. In this case, the FL system accepts the test error only if it improves the global model and rejects it if it does not enhance the model. This proactive defense mechanism in the FL system explicitly detects erroneous model updates and prevents impacts. Although this form of defense works well against un-targeted attacks, it does not produce the same results as targeted adversarial attacks [31]. Training the backdoor data usually generates a poisoned model that behaves like the intact model. To mitigate this limitation, Wang et al. [32] proposed the AUROR protocol, a defense mechanism that leverages k-means clustering algorithm to detect harmful node updates in targeted attacks. The AUROR protocol isolates all the nodes participating in the FL learning process into many clusters based on their uploaded features. For every feature, the protocol generates clusters of benign and malicious nodes. However, despite the nobility of this protocol, it does not address the challenge of high-dimensional big data for the model weights [33].

1.2.9. Differential Privacy

While initially conceived as a defense mechanism against privacy attacks, differential privacy is increasingly finding its use case in mitigating security attacks. Differential privacy can help prevent data poisoning attacks by injecting random noise into the model updates [24] by injecting the model updates with statistical noise, and differential privacy achieves two things.

First, it guarantees that no dataset can validly be distinguished from the rest of the nodes' local data samples. Second, an adversary that modifies a few training samples cannot cause a significant deviation in the overall global model. The main problem with differential privacy is that the statistical noise added to the model updates is injected into the learning algorithm's noise. The accumulated noise can compromise the overall global model when it exceeds the limit [34,35]. Besides the accumulated noise, differential privacy can also not work when the FL system has many nodes [36].

Various studies have proposed algorithms to robustly detect faulty updates and protect FL systems from malicious training [19]. Most research in this area has primarily focused on protocols that can sustain communication instabilities, faulty model updates, and node dropouts. One such protocol is the adaptive aggregation that combines repeated median and reweighting approaches in iteratively least squares [37]. This method has demonstrated robust security against model corruptions prevalent with free-riding attacks [9].

Another often cited method is the Gaussian distribution that determines nodes' potential contributions to the global model [18]. This model leverages layer-wise optimization steps to work effectively on differential functional units in DNNs and tackle heterogeneity problems in data. Despite the novelty of these approaches, secure aggregation protocols have a primary challenge. They can leave other attacks, such as man-in-the-middle attacks, unnoticed because of the FL's decentralized nature. Pruning is quickly emerging as a defense mechanism against security attacks in FL systems that want to minimize the magnitude of the global model [20]. By minimizing the complexity of the model updates, pruning helps FL systems enhance their overall accuracy. The protocol assumes that a decentralized FL system has many nodes with limited bandwidth and computational power. Another often cited method is the Gaussian distribution that determines nodes' potential

contributions to the global model [18]. This model leverages layer-wise optimization steps to work effectively on differential functional. As nodes cannot effectively train a large-sized DNN, but with pruning, this limitation is avoided by leaving out small local datasets that do not require training. There are two common forms of pruning: Federated Dropout (FD) and PruneFL. In FD-based learning, nodes train their local dataset on smaller subsets of the overall global model. FD helps to avoid the problem of local computation and low bandwidth. PruneFL, on the other hand, maximizes the estimated empirical risks involved in training small datasets [38,39]. The past few years have seen ZKPs emerging as privacy-preserving mechanisms that allow data to be verified without revealing that data [40]. ZKPs use cryptographic primitives to validate statements by one entity (called the prover) to another entity (called the verifier) without sharing their underlying data in their most basic form [41].

The past few years have seen ZKPs emerging as privacy-preserving mechanisms that allow data to be verified without revealing that data [40]. ZKPs use cryptographic primitives to validate statements by one entity (called the prover) to another entity (called the verifier) without sharing their underlying data in their most basic form [41]. The primary goal of ZKPs is to validate a statement without necessarily leaking any extra information [39]. In FL systems, ZKPs can provide an efficient solution for verifiability challenges on personal data. Because they avoid sharing nodes' private data, leveraging them ensures that clients only submit model updates with pre-specified characteristics. This can help the learning system defend against backdoor attacks and model corruption. For example, ZKPs can verify that model updates were trained correctly by data instances from smartphones without necessarily sharing their private data. ZKPs have compelling value propositions in FL systems as defense mechanisms against sharing private data. However, they are merely probabilistic statements and cannot provide complete certainty that is verifiable [9]. Their application is still in its infancy stages, which means they lack standards that specify how they should be used. Adversarial training has emerged as one of the most robust defense techniques against the DNN model adversaries. Unlike other strategies that focus on extrinsic factors, adversarial training concentrates on intrinsically improving the DNN model resilience. When applied in FL systems, adversarial training solves the minimax optimization problem in two ways. First, the inner maximization generates adversarial examples by optimizing the classification loss. Second, the outer minimization yields model parameters by minimizing the loss of adversarial datasets from internal maximization [42]. This way, adversarial training can provide resilience against common threats such as evasion and backdoor.

Exchanging model parameters under limited bandwidth is too costly, especially with large-scale DNNs. FD has a compelling value proposition in such environments because it only transmits model results whose dimensions are relatively smaller than the model sizes. One FD variation that has gained popularity in the academic community over recent years is knowledge distillation (KD). With KD, knowledge is transferred from a fully trained model (also called the teacher model) to a smaller model (known as the empty student model). Such knowledge can also aggregate other students' model updates that form in the FL system. The primary objective of KD-based systems is to enhance the robustness of FL systems by minimizing computational and bandwidth costs [43]. Table 1 states defense mechanisms against security attacks in FL systems and their limitations.

Table 1. Summary of defence mechanisms against security attacks in FL systems and their limitations.

Defence Mechanism	Security Attacks	Limitation (s)
Anomaly detection	Data poisoning attacks Model poisoning attacks Free-riding attacks	The test error rate method can generate a poisoned model that appears and behaves as the intact models. It does not address the problem of high-dimensional big data for the model weights.
Differential privacy	Data poisoning attacks Model poisoning attacks Inference attacks	The accumulated noise from the model updates and the learning algorithm can compromise the overall global model when it exceeds the limit. The protocol cannot work effectively with large decentralized nodes in FL systems.
Secure Aggregation	Data poisoning attacks Model poisoning attacks Backdoor attacks	It can leave other attacks to go unnoticed because of the FL's decentralized nature.
Pruning	Backdoor attacks	It cannot work with large decentralized nodes in FL systems
ZKPs	Data poisoning attacks Model poisoning attacks Backdoor attacks Man-in-the-middle attacks	ZKPs merely provide probabilistic statements that cannot provide complete certainty that is verifiable. There are no standards that organizations can use to implement ZKPs.
Adversarial Training	Inference attacks	The protocol cannot work effectively with large decentralized nodes in FL system.
Federation Distillation	Inference attacks Man-in-the-middle attacks GANs attacks	It does not address the problem of high-dimensional big data for the model weights

2. Methodology

This section introduces our proposed system. Our framework facilitates FL by leveraging Ethereum's incentive mechanisms and proof-of-work (PoW) consensus algorithm to promote trust among the decentralized nodes.

2.1. System Model

We consider a classification problem using the popular Fashion-MNIST dataset. This image classification problem consists of 60,000 training samples and 10,000 test data samples [43]. Each example is a 28×28 grayscale image with a label from 10 classes. In our case, the input is a onechannel ($1 \times 28 \times 28$) pixel with values [0, 1] while the output is a 10-class label with values [1, 2, . . . , 10] as follows:

$$\left\{ inMNIST \in R^{1*28*28} \,\middle|\, 0 < inMNIST < 1 \right\} \tag{2}$$

$$\{ outMNIST \in Z | 0 < 1 < outMNIST < 10 \} \tag{3}$$

We regard each problem as a set of test dataset tuples where each tuple represents {P; BlockFL; Y_n} has $P = \sum_{I=1}^{m} inMNIST$, BlockFL is the Blockchain-enabled FL function, and Y_n is the outputwhose value is given as as $Y_n = f(x_n)$ on the test data set for a vector x_n. Each node in the BlockFL system has an associated key pair (pub_{key}, $priv_{key}$).

This paper aims to leverage the Fashion MNIST data and use it as a basis for collaboratively training the DNN model. Because our approach uses a homogeneous dataset, we will not perform a normalization process. However, for heterogeneous data sets, normalization is required to obtain better results. We will leverage a Blockchain-powered framework to train them and share the collaborative model. In our case, the main goal of FL will remain the same even when powered by blockchain combining the weights of a locally trained model [44]. After aggregating the locally trained model, the global model gets stored on a Blockchain-enabled system. The security and privacy of data of the decentralized nodes is the main reason we are considering blockchain in this paper. Essentially, blockchain manages the privacy and leakage of private data helping to prevent various attacks. Therefore, the ledger will store two kinds of transactions: data sharing transactions and recovered transactions. To mitigate against security attacks and ensure privacy, we will use a public permissionless blockchain. Permissionless Blockchains such as Ethereum allow any node to append transactions to the ledger provided they have enough tokens.

We also believe collaborative learning can thrive if all the nodes participate in the training process. The proposed Blockchain-enabled FL system combines local model weights and transmits them as data sharing transactions to the global model. Local nodes that want to access the global model downloads it as data recovered transactions. The proposed system has two components local model and Blockchain-enabled FL, as shown in Figure 1.

2.2. CNN Classification Model

Image classification is one of the fundamental topics in DNNs. Li and Wang researching the subject [44] have demonstrated remarkable performances with image classification tasks with the development of convolutional neural network (CNN) architecture. We will use principal component analysis (PCA) [45] to reduce the dimensionality of the Fashion MNIST dataset. To achieve greater levels of abstraction, the CNN will have four layers input layer, pooling layer, fully connected layer, and output layer. We model each component of the CNN as a vector V_i rotated and translated through a weighted matrix $M_{i,j}$ to vector $\vec{v}_{i|j}$. We can now calculate the prediction vector as follows:

$$\vec{v}_{i|j} = M_{i,j} \, u_i \tag{4}$$

The next higher layer, i.e., N_i computes the sum of all the predictions from lower-level layers with $c_{i,j}$ as coupling coefficients. N_i can be represented as follows:

$$N_i = \sum_i c_{i,j} \vec{v}_{(i|j)} \tag{5}$$

where $c_{i,j}$ is simply the routing softmax function specified as:

$$c_{i,j} = \frac{e^{bij}}{\sum_k e^{bik}} \tag{6}$$

Prediction vector calculation based on higher level sum in Algorithm 1 lower layer denoted as l, vector $\vec{v}_{i|j}$, M Matrix and this algorithm performs coefficients. We now specify the routing algorithm as follows:

Algorithm 1 Component Analysis

1: For i in layer l and i in layer $(l + 1)$
2: $b_{i,j} \leftarrow 0$
3: For k repeats do:
4: Forall i in layer l do $c_{i,j}$
5: Forall j in layer $(l + 1)$ do N_i
6: Forall j in layer $(l + 1)$ do N_i Iterate
7: i in layer l, j in layer $(l + 1)$ do
8: $b_{i,j} \leftarrow b_{i,j} + c_{i,j} \vec{v}_{i|j}$
9: Return $\vec{v}_j$

2.3. Local Model and Blockchain Proposed Model

The global model is stored on a Blockchain-enabled system after the locally learned models have been aggregated. The main purpose of Blockchain in this article is to ensure confidentiality and privacy of data held by decentralized nodes. In essence, blockchain controls the confidentiality and leakage of private data, preventing a variety of threats.

2.4. Consensus Mechanism

A consensus algorithm is a method by which all peers in a blockchain network reach consensus on the current state of the distributed ledger. Consensus algorithms provide resilience to the blockchain network and in this way establish trust between unknown peers in a distributed computing environment. Essentially, the consensus protocol ensures

that each new block added to the blockchain is the only version of the truth that all nodes in the blockchain agree on. The blockchain consensus protocol has several specific goals, including reaching agreement, collaboration, cooperation, equal rights for all nodes and calculations for each node.

2.5. Reduction of Dimensionality

In the case of this model, the process is codified and obtained through principal component analysis. Since our data have high dimensionality, we will resort to a type of dimensionality reduction by feature selection, principal component analysis. We will first calculate 40 principal components and determine the number of components we will use in our calculations.

Figure 2 illustrate the plot, we achieved from running Algorithm 2 which presented the results as 80% of variance is captured between the before the 25th PC this provides information about the presence of the counted components are insufficient. We worked on Plotting Accuracies and Losses techniques and in this model CNN accuracies and losses were plotted to determine the training and validation loss and training and validation accuracy of the model.

Algorithm 2 Component Analysis

1: **import** tensorflow **as** tf
2: **from** tensorflow **import** keras
3: **import** numpy **as** np
4: **import** matplotlib.pyplot **as** plt
5: **from** sklearn.decomposition **import** PCA
6: **from** sklearn.ensemble import RandomForestClassifier **from** sklearn.metrics **import** accuracy_score
7: **from** sklearn.pipeline **import** Pipeline
8: **from** tensorflow.keras.utils **import** to_categorical from tensorflow.keras.models **import** Sequential
9: **from** tensorflow.keras.layers **import** Dense, Flatten, Conv2D, Dropout, MaxPooling2D **from** sklearn.model_selection **import** GridSearchCV, ShuffleSplit, train_test_split **from** sklearn.svm **import** SVC
10: **from** sklearn.linear_model **import** LogisticRegression **from** sklearn.neighbors **import** KNeighborsClassifier **from** sklearn.neighbors **import** NearestCentroid **from** sklearn **import** metrics
11: **from** sklearn.cluster **import** KMeans
12: **from** sklearn.metrics **import** classification_report fashion_mnist
= tf.keras.datasets.fashion_mnist
13: (train_images, train_labels), (test_images, test_labels) = fashion_mnist.load_data
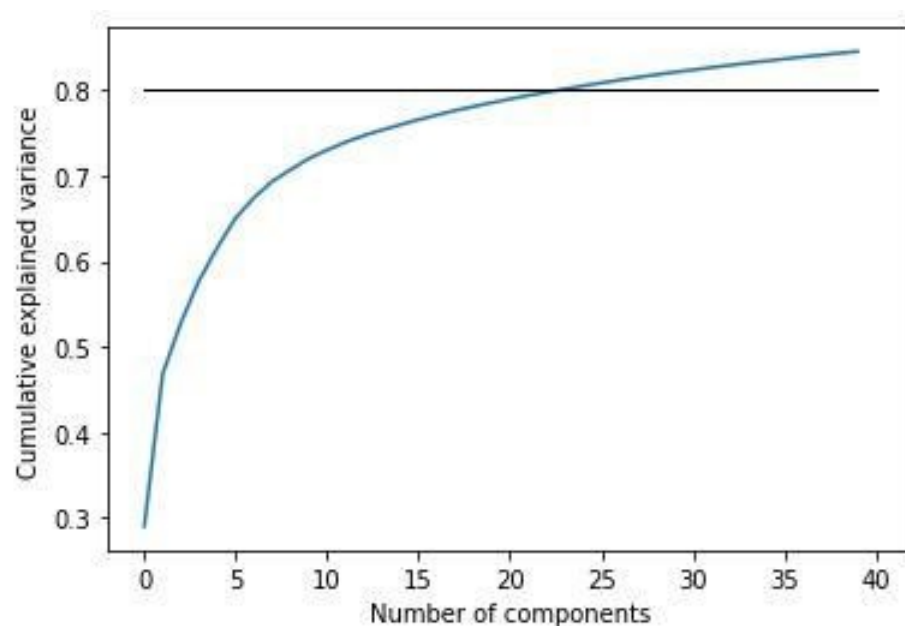

Figure 2. CNN-Components Calculation.

In Algorithm 3, we summarized the analysis of components that presented Results performing well in validation accuracy from the beginning hovering around 0.90. We can see that the dataset becomes learned around the 15th epoch. Figure 3 illustrates the analysis of the CNN-training and Validation accuracy plot, and Figure 4 the CNN-Training and Validation Loss exemplify.

Algorithm 3 Component Analysis

1: Cnn_accuracy = Cnn_model_training.history['acc']
2: Cnn_val_accuracy = Cnn_model_training.history['val_acc']
3: loss = Cnn_model_training.history['loss']
4: val_loss = Cnn_model_training.history['val_loss']
5: epochs = [i for i in range (1,51)]
6: plt.plot(epochs, cnn_accuracy, 'b–', label = 'Test accuracy')
7: plt.plot(epochs, cnn_val_accuracy, 'r', label='Validation accuracy')
8: plt.title('Training and validation accuracy')
9: plt.savefig('Accuracy values')
10: plt.legend()
11: plt.figure()
12: plt.plot(epochs, loss, 'b–', label = 'Training loss')
13: plt.plot(epochs, val_loss, 'r', label = 'Validation loss')
14: plt.title('Training and validation loss')
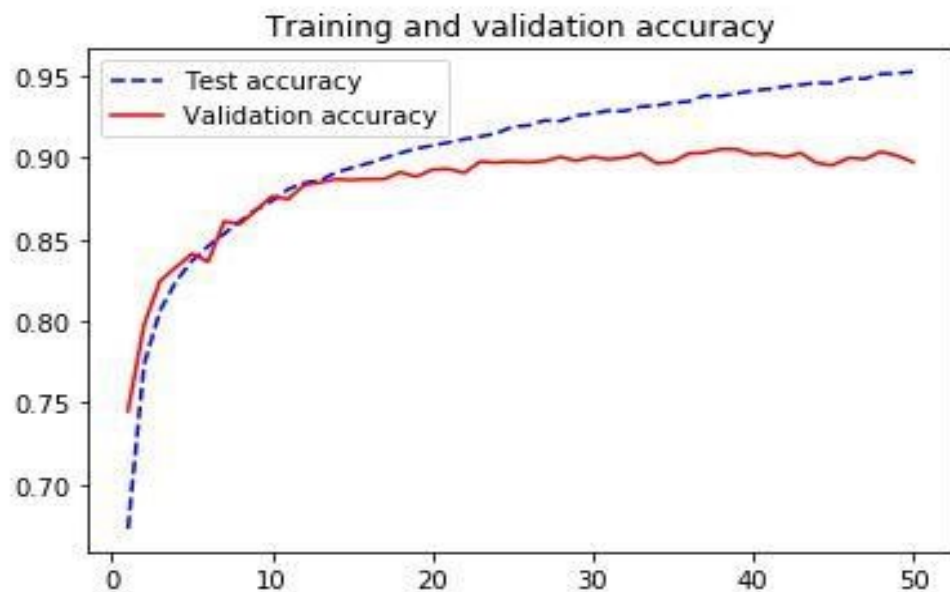15: plt.legend()
16: plt.savefig('Loss values')
17: plt.show()

Figure 3. CNN-Training and Validation Accuracy.

2.6. Multinomial Logistic Regression

Multinomial logistic regression is another method used to analyze our data. We apply the lasso and regularization techniques and compare the outcomes. It was done on both primary and secondary components. In Algorithm 4 we put 25 PCs, we'll employ a pipeline technique. The One-Vs-Rest strategy will be employed, which trains K binary classifiers, where K is the number of classes. Pipeline is a sklearn technique that allows us to combine data modifications and the final estimator in one step.
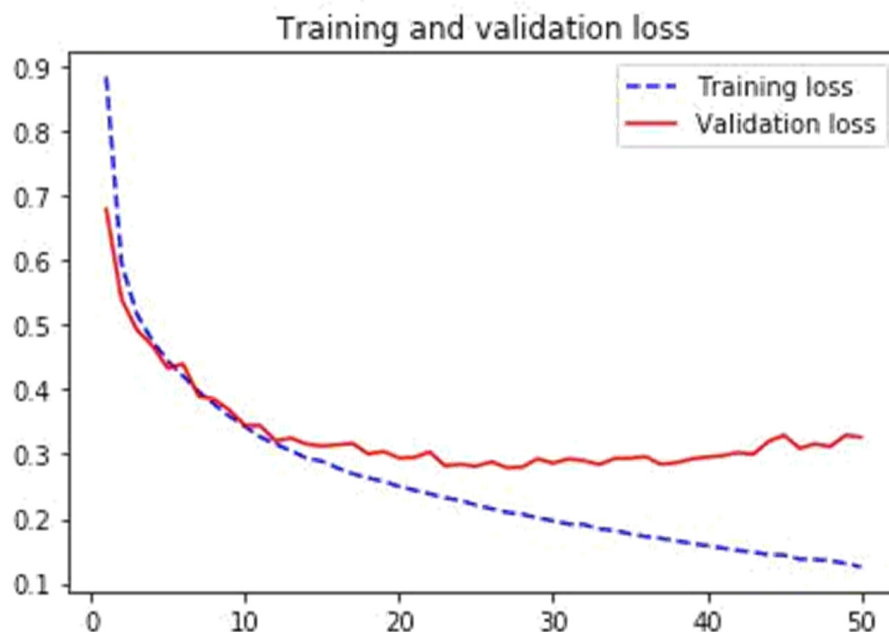
Figure 4. CNN-Training and Validation Loss.

Algorithm 4 Multinomial Logistic Regression

1: pipe_log_reg_l2=Pipeline([('pca',PCA(n_components=25)),
('clf',LogisticRegression(multi_class='ovr',penalty='l2',solver='saga'))])
2: pipe_log_reg_l2.fit(train_images_mod,train_labels)
3: pred_labels_pipe_log_reg_l2 = pipe_log_reg_l2.predict(test_images_mod)
4: test_acc_pipe_l2 = accuracy_score(pred_labels_pipe_log_reg_l2,test_labels)
5: print('Test accuracy of logistic regression on 25 PCs with l2 penalty : ' + str(test_acc_pipe_l2))

2.7. Supports Vector Machines

Unlike other models, the proposed model supports the vector machine. It is limited to the use of PCs because of the high dimensionality of the data. Below, Algorithm 5 illustrates the Model accuracy with a polynomial kernel. We choose to use 25 PCs because we believe we can get good results with this variance. The polynomial kernel and the RBF (Gaussian) kernel were used.

Algorithm 5 Models Accuracy

1: pipe_svm_1 = Pipeline([('pca', PCA(n_components=25)),('clf',SVC(C=100,kernel='rbf',gamma=0.1))])
2: pipe_svm_1.fit(train_images_mod,train_labels)
3: pipe_svm_1_pred_labels = pipe_svm_1.predict(test_images_mod)
4: pipe_svm_1_acc = accuracy_score(pipe_svm_1_pred_labels,test_labels)
5: print('The accuracy of SVM classifier with rbf kernel is equal to :' + str(pipe_svm_1_acc))
6: pipe_svm_2 = Pipeline([('pca', PCA(n_components=25)),('clf', SVC(C=100,kernel='poly',degree=5))])
7: pipe_svm_2.fit(test_images_mod,test_labels)
8: pipe_svm_2_pred_labels = pipe_svm_2.predict(test_images_mod)
9: pipe_svm_2_acc = accuracy_score(pipe_svm_2_pred_labels,test_labels)
10: print('The accuracy of SVM classifier with polynomial kernel of degree five is equal to :' +
str(pipe_svm_2_acc))

2.8. Blockchain-Enabled FL

Traditional DL systems provide no guarantees regarding the integration of the aggregation process and rely on trust instead. These designs are also prone to single-point-of-failures that by design extend to centralized platforms. This paper proposes a multi-layered secure FL platform powered by Blockchain. The decentralized training nodes require motivation to provide computational resources and provide data. A transparent incentive mechanism backed by smart contracts can help an FL system offer a financial incentive to

decentralized nodes and encourage good behaviors on the network. A Blockchain-enabled FL guarantees security and system availability by decentralizing trust. The application of Blockchain in FL systems has received widespread use cases in healthcare. Privacy and security [46] and the internet of things (IoT) [21]. Authors Wang et al. in these studies, has proposed using public and permissionless Blockchains such as Ethereum [22], while others believe that private, permissioned Blockchains like HyperLedger Fabric could be considered for FL [47]. The proof-of-work (PoW) consensus algorithm in Blockchains is increasingly facing scalability challenges, making it redundant for efficient processing in FL systems [48,49]. Consequently, Kumar et al. [50] have recommended the use of other scalable-friendly consensus protocols such as proof-of-stake (PoS) and proof-of-authority (PoA). Mikolajczyk and Grochowski [47] proposed a Blockchain-powered FL system that leverages up-to-date data from decentralized hospitals to enhance the recognition and privacy of computed tomography (CT) images. This model uses permissioned Blockchains in FL to detect COVID-19 patients using a lung screening approach where hospitals serve as decentralized nodes. Rajalakshmi and Tuhina [51] and Xie et al. [49] discuss various segments of a Blockchain-powered privacy-preserving FL network. Their frameworks allow them to learn and share private users' data via a federated architecture that encrypts the data before training the globally coordinated model update. Finally, Pop et al. [18] proposed permission for Blockchain-enabled FL to detect intrusions in IoT systems. Their model, which used Multichain, enhanced audibility and transparency with a negligible runtime overhead of between 5% to 15%. This paper differs from the previous works in several aspects. First, previous studies such as [43,52,53] have focused primarily on how Blockchain can implement FL systems. Since Blockchain is still in its infancy stages, these studies provide a framework on how FL systems can benefit from Blockchain and provide security. In doing so, these studies have only provided a general framework that lacks an incentive mechanism. While their approach advances knowledge in this field, it leaves out a crucial component of incentivization. This paper bridges this knowledge gap by proposing a transparent incentivization mechanism that can promote good behavior among the participating decentralized nodes and avoid common issues such as poisoning and backdoor attacks.

2.9. Proposed Blockchain Model

The Fashion MNIST dataset is large and requires high-capacity storage. Placing such a dataset on a Blockchain with a limited storage capacity is expensive, both in computational resources and financially. Therefore, the actual Fashion MNIST dataset will be stored locally on the decentralized nodes, while Blockchain will only store the global model updates and help the clients retrieve the trained model. In this regard, each node stores a transaction in the block while the Fashion MNIST training data remains local to the client. This allows multiple nodes to collaboratively share the Fashion MNIST data and train the FL model for optimal results. Since the nodes leverage Blockchain to transmit the transactions to and from the centralized server, this mechanism does not violate the privacy of nodes. The Kademlia protocol proposed by Maymounkov and Mazieres can provide a framework for a multi-organization architecture that enhances privacy in trustless, decentralized networks [27]. Using this protocol, we partition all nodes, N, into categories to share data transparently and securely. Each category belongs to a different community that maintains a log table, specified as. To retrieve the Fashion MNIST dataset from the physically present nodes, we use the Kademlia's Kademlia XOR mechanism that defines the distance nodes below:

$$d_i(N_i, N_j) = \frac{\Sigma p, q \in \left\{ N_i \cup N_j - N_i \cap N_j \right\} \left(x_{pq}^{N_i} + x_{pq}^{N_j} \right)}{\Sigma p, q \in \ N_i \cup N_j \left(x_{pq}^{N_i} + x_{pq}^{N_j} \right) \cdot \log \left(d_p \left(N_i, N_j \right) \right)} \tag{7}$$

where N is the data categories to access the data from decentralized nodes, $d_i(N_i, N_j)$ is the distance of the two nodes, N_i and N_j; $\left\{ N_i \cup N_j - N_i \cap N_j \right\}$ is the Kademlia's XOR

metric, and $x_{pq}^{N_i} + x_{pq}^{N_j}$ are the attributes for the weight matrix of the two nodes, N_i and N_j Blockchain stores all each node's unique identifiers (IDs) based on the logic and distance of the decentralised nodes. For the two nodes with $N_i(ID)$ and $N_j(ID)$, the equation below represents the distances between them:

$$d_i(N_i, N_j) = N_i(ID) \oplus N_j(ID) \tag{8}$$

To achieve a privacy-preserving mechanism on the decentralized network, we use the randomized method for two nodes shown below:

$$N_r[A(R)\epsilon\, S] \leq exp\,(\varepsilon) \cdot N_r[A(R')\epsilon\, Z] \tag{9}$$

where R and R' are the adjacent records of data, Z is the outcome set of data, while $N_r[A(R)\epsilon\, S]$ is the privacy-preserving function of the data in the decentralised network. For multiple nodes, we use the Laplace function in the local model training (m_l) as follows:

$$\vec{m}_l = m_l + Laplace\left(\frac{s}{\varepsilon}\right) \tag{10}$$

where s is the sensitivity of the privacy-preserving function described by the equation below:

$$s = max_{N,N'} f(N)f(N') \tag{11}$$

To ensure a data privacy-preserving mechanism, we will encrypt all the data via a public key infrastructure (PKI) [54] comprising of two keys: a public key (Pub_{key}) and a private key (Pri_{key}). When all the transactions get validated, they are appended to the Blockchain. In Algorithm 6 the Consensus Protocol presenting the blocks validations.

Algorithm 6 Consensus protocol

1: $m_i \leftarrow N_j : N_i$.
2: *leader* $\leftarrow m_i : N_j$
3: $N_i, N_j \leftarrow$ *leader*
4: Validate N_i, N_j
5: *Ledger* $\leftarrow$ *Blocks*

Current approaches leverage the PKI system to protect data, which to some extent provides data protection. However, this is not enough considering the emerging sophisticated attacks and the increasing computational power. Our solution does not share the original training data from decentralized nodes. Instead, it only exchanges locally learned model weights with the global model. Local nodes that want to access the global model downloads it as data recovered transactions. The objective of this architecture is to train the global model via locally trained model and achieve an FL mechanism that preserves data privacy with multi-layered security. In Algorithm 7 Privacy-preserving flow chart explained and graphically summarized in Figure 5.

Algorithm 7 Privacy-preserving FL flow chart

1: For *Each node n*
2: For *all N Decentralized node*
3: For *all i iterations*
4: *While (Accuracy, Privacy)*
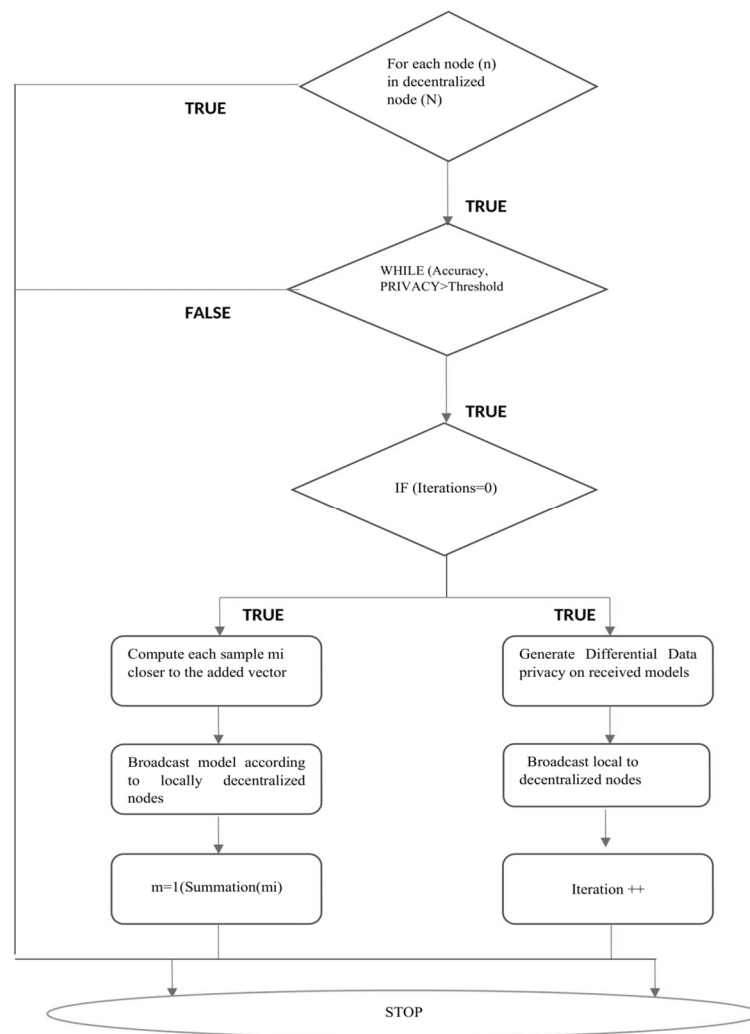5: *For m = 1*
6: Return m_i

Figure 5. Privacy-preserving FL flow chart.

3. Result and Discussion Conclusions

Our proposed system comprises two components the FL system and the underlying Blockchain.

3.1. Environment

We implement an FL environment using Python (version 3.9.0) with the following libraries:

- Keras. This is a Python and R-based DL library.
- TensorFlow. This is an open-source library for differentiable programming in multiple
- DL tasks.
- NumPy. This is a Python-based library for computing high-level mathematical functions and multi-dimensional arrays and matrices.
- Matplotlib. This is a plotting library in a Python programming environment that extends NumPy.

Our framework will use the Fashion MNIST as the dataset. We extracted 70,000 samples in 10 categories and then split the dataset into 60,000 examples for the training dataset and 10,000 examples in the test dataset. We used a low-resolution scale (28 × 28) in grayscale, vital for appropriate segmentation. We will use 10 decentralized nodes (N = 10) and one central aggregator and run the model in 10 rounds by default. The test data set is used only for evaluation purposes and is not included in any nodes' train dataset.

To simulate a data poisoning attack with N decentralized nodes where x% are malicious, we randomly designate (nodes as malevolent from the set N at the start of each experimental round. For example, if 10% of the nodes are malicious and N = 10, we have () malicious nodes). We then consider three label attack settings that represent diverse conditions (1) source class/target class pairing where the source class gets frequently misclassified (2), a pairing system where the source class is infrequently classified as the target class and (3) a pairing system involving two extremes.

For this data set, we experiment with:

1. A shirt (label 6) gets paired with a t-shirt/top (label 0).
2. A trouser (label 1) gets paired with a dress (label 3).
3. A coat (label 4) gets paired with a shirt (label 6).

After implementing the FL system, we proceed to develop the Blockchain system. Whenever a locally trained update gets submitted to the aggregator, the aggregator submits it to decentralized nodes for verification. During the verification, all the participating nodes verify that the submitting node has enough tokens in its account and has followed all the procedures required to generate a new local update. When these nodes agree, the global weight gets updated, and the aggregator broadcasts it to the decentralized nodes as its new model update. We use Ethereum as an underlying Blockchain to provide privacy-preserving capabilities. When a validator block gets committed, the submitting node receives ERC-20 tokens as incentives. A wallet address represents the identity of each decentralized node on the Ethereum Blockchain is represented by a wallet address (a combination of (Pub_{key}) and (Pri_{key}) that is managed by MetaMask [55]. Each time a validator updates the global model weight, its position is recorded on Ethereum. The system then computes ERC-20 tokens sent to the node's wallet address as an incentive.

3.2. Simulation Process

We consider two scenarios:

Scenario 1: We assume a conventional FL system (without Blockchain) has N decentralized nodes of which m% are malicious. In this case, the malicious node can be a data poisoning or a model poisoning agent. We randomly designate $N_m = N * m/100$, where N_m represents the malicious actors at the beginning of each experiment. The rest of the nodes ($N - N_m$) are honest participants. We repeat the experiment 10 times and use the average result to address the impact of random selection for malicious nodes. We tweak the value m from 2% to 50% in the study.

Scenario 2: We assume a Blockchain-enabled FL system built atop the Ethereum network as described earlier with three smart contracts: training, aggregation, and completed. The intelligent training contract provides a framework where decentralized nodes submit the local update processes. When the last required update gets submitted, the smart aggregation contract is triggered. This allows the trainers to submit their local aggregation output addresses. When the last required aggregation output address is transmitted, the consensus protocol selects one aggregation address that becomes the new global weight address. If this is the last training iteration, the contract transitions to completed. Otherwise, the system goes back to the training stage and starts over with the following training round. As with scenario 1, we will repeat the experiment 10 times and report the average result. We assume each training node is a single core machine with 4 GB of RAM and running Ubuntu 18.04 LTS. To cut down on gas fees and testing purposes, each node does not run its Ethereum node. Instead, each decentralized node connects to the Infura network via the Rinkeby Testnet. This way, nodes do not have to synchronize to the Ethereum main net fully.

4. Findings

We begin by examining the feasibility of data in a conventional FL system. Figure 6 shows the global model accuracy and source class in cases with m ranging from 2% to 50%.

We averaged the results for 10 rounds for each setting of m%. The dark black bars denote the mean, while slim black bars at top denote the standard deviation.
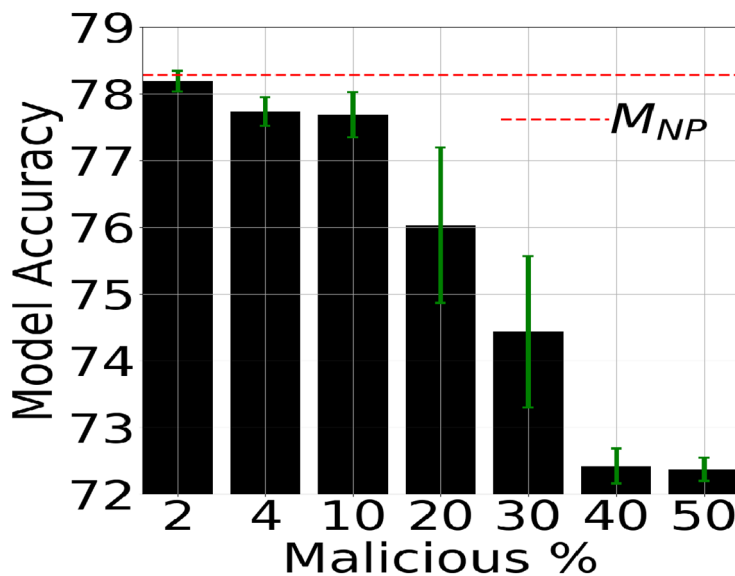

Figure 6. Attack visibility and impact of malicious nodes on a conventional FL system.

The finding shows that the test accuracy (global model utility) decreases as the malicious nodes increase. Even with small m, there is a corresponding decrease in test accuracy with an even large source recall compared to a non -poisoned system (denoted by MNP) in the figure. For this dataset, we considered a loss in source class recall for the three *source → target* class settings where (1) shirt (label 6) gets paired with a t-shirt/top (label 0), (2) a trouser (label 1) gets paired with a dress (label 3), and (3) a coat (label 4) gets paired with a shirt (label 6). Table 2 shows *source → target* with different counts for the Fashion-MNIST dataset.

Table 2. Loss in *source → target* with different counts for the Fashion-MNIST dataset.

$\mathbf{FM_{source}} \rightarrow \mathbf{FM_{target}}$	Percentage of Malicious Nodes						
	2	**4**	**10**	**20**	**30**	**40**	**50**
$6 \rightarrow 0$	0.03	0.34	2.15	3.32	23.48	34.17	47.51
$1 \rightarrow 3$	0.67	0.64	13.23	23.25	31.23	36.35	46.25
$4 \rightarrow 6$	0.00	2.34	5.17	10.79	25.67	33.25	43.21

The findings show that even a small proportion of malicious nodes in the FL system can impact the global model utility. For example, when m = 4%, we observe that the source class recall drops by 2.34%, misclassifying a coat as a shirt. When m increases to 50%, we note that the source recall class decreases by a staggering 47.51%, misclassifying a shirt as a t-shirt/top. This means that an adversary that controls even small proportion of the total decentralised nodes can significantly impact the overall global model utility. Next, we investigate the feasibility of data and model poisoning attacks under a Blockchain-enabled FL system. Table 3 shows *source → target* with different counts for the Fashion-MNIST dataset when the experiment is conducted under a Blockchain -enabled FL platform.

Table 3. Loss in *source* → *target* with different counts for the Fashion-MNIST dataset under Blockchain-Enabled FL.

$FM_{source} \rightarrow FM_{target}$	Percentage of Malicious Nodes						
	2	**4**	**10**	**20**	**30**	**40**	**50**
$6 \rightarrow 0$	0.00	0.00	0.00	0.00	0.04	3.24	5.84
$1 \rightarrow 3$	0.00	0.00	0.00	0.00	0.56	1.78	4.56
$4 \rightarrow 6$	0.00	0.00	0.00	0.00	0.78	6.25	7.24

From the table, the platform achieves higher security when compared to a conventional FL system. Blockchains are inherently resilient to integrity attacks such as the data or model poisoning that we have investigated. The only option that an adversary would attack such a network would be by attacking the communication to the trainer nodes. If the adversary were to succeed in hijacking and gaining complete control over the training operations, it would tamper with the overall aggregation mechanism. In this case, submitting a faulty aggregation would result in a different Ethereum address that the honest nodes would outrightly ignore because of a consensus protocol. This explains why Blockchain achieves higher security in FL systems. However, when m increases beyond 30%, the model becomes susceptible to some form of percentage attacks. This is attributable to the nature of decentralized learning, where the order in which model updates arrive at the aggregator is not necessarily the same as they would be validated [18,42]. The increase in the attack percentage would most likely arise from other attacks other than poisoning threats.

5. Conclusions

In this paper we showed that poisoning attacks (data and model poisoning attacks) are inherent in FL systems and can significantly affect the accuracy of the global model utility. We demonstrated that an increase in adversaries negatively impacts the global model utility, and it is possible to achieve targeted poisoning impacts. We also showed that an incentive based Blockchain mechanism could minimize poisoning attack surfaces in FL systems. While this study shows how an incentive mechanism helps Blockchain-powered FL systems provide multi-layered security, it has a weakness: it assumes the dataset is homogeneous. The proposed Blockchain enabled FL system aggregates data from the local model and sends them to the global model as data sharing transactions. The proposed model FL is facilitated by our architecture, which makes use of Ethereum's incentive mechanisms and proof-of-work (PoW) consensus algorithm to foster cooperation across decentralized nodes. In which blockchain was considered because of the security and privacy of data and the decentralized nodes. The proposed Blockchain-enabled FL system aggregates local model weights and sends them to the global model as data sharing transactions. The global model is downloaded as data retrieved transactions by local nodes that desire to access it. Local model and Blockchain-enabled FL are the two components of the proposed system Blockchain-enabled FL.

- Accuracy is improved.
- Privacy and security are improved.
- The proposed Blockchain-enabled FL system combines local model weights and transmits them as data sharing transactions to the global model.
- In our proposed model, we can see a 95% Accuracy test after plotting the Accuracy test compared to other models above.

Future studies can be undertaken to examine incentive-based Blockchains' impact on FL systems that have asynchronous and heterogeneous training datasets. Federated Learning and challenges facing federated learning was addressed. Data Poisoning in Federated learning and Model Update on Data Poisoning was addressed. The study also relied on Infura and Rinkeby Testnet to minimize gas fees, which means that the decentralized

nodes did not fully synchronize with the main chain. Future studies should ascertain the true impact of an incentive-based Blockchain-enabled FL system under practical scenarios involving the mainnets.

References

1. Wang, J.; Cao, B.; Yu, P.; Sun, L.; Bao, W.; Zhu, X. Deep Learning Towards Mobile Applications. In Proceedings of the IEEE 38th International Conference on Distributed Computing Systems (ICDCS), Vienna, Austria, 2–6 July 2018.
2. Roess, A. The Promise, Growth, and Reality of Mobile Health-Another Data-Free Zone. *N. Engl. J. Med.* **2017**, *377*, 2010–2011. [CrossRef] [PubMed]
3. Yao, X.; Huang, T.; Wu, C.; Zhang, R.; Sun, L. Towards Faster and Better Federated Learning: A Feature Fusion Approach. In Proceedings of the IEEE International Conference on Image Processing (ICIP), Taipei, Taiwan, 22–25 September 2019.
4. Tuttle, H. Facebook Scandal Raises Data Privacy Concerns. *Risk Manag.* **2018**, *65*, 6–9.
5. Cheng, Y.; Liu, Y.; Chen, T.; Yang, Q. Federated Learning for Privacy-Preserving AI. *Commun. ACM* **2020**, *63*, 33–36. [CrossRef]
6. Truong, N.; Sun, K.; Wang, S.; Guitton, F.; Guo, Y. Privacy Preservation in Federated Learning: An Insightful Survey from the GDPR Perspective. *Comput. Secur.* **2021**, *110*, 102402. [CrossRef]
7. Bouacida, N.; Mohapatra, P. Vulnerabilities in Federated Learning. *IEEE Access* **2021**, *9*, 63229–63249. [CrossRef]
8. Tolpegin, V.; Truex, S.; Gursoy, M.E.; Liu, L. Data Poisoning Attacks Against Federated Learning Systems. In Proceedings of the European Symposium on Research in Computer Security, Darmstadt, Germany, 14–18 September 2020.
9. Nasr, M.; Shokri, R.; Houmansadr, A. Comprehensive Privacy Analysis of Deep Learning: Passive and Active White-Box Inference Attacks Against Centralized and Federated Learning. In Proceedings of the IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 20–22 May 2019.
10. Ma, J.C.; Li, J.; Ding, M.; Shi, L.; Wang, T.; Han, Z.; Poor, H.V. When Federated Learning Meets Blockchain: A New Distributed Learning Paradigm. *Comput. Sci. Netw. Internet Archit.* **2020**, *2009*, 09338.
11. ur Rehman, M.H.; Salah, K.; Damiani, E.; Svetinovic, D. Towards Blockchain-Based Reputation-Aware Federated Learning. In Proceedings of the IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Virtual Conference, 2–5 May 2020.
12. Mach, P.; Becvar, Z. Mobile Edge Computing: A Survey on Architecture and Computation Offloading. *IEEE Commun. Surv. Tutor.* **2017**, *19*, 1628–1656. [CrossRef]
13. Yousefpour, A.; Fung, C.; Nguyen, T.; Kadiyala, K.; Jalali, F.; Niakanlahiji, A.; Kong, J.; Jue, J.P. All one needs to know about fog computing and related edge computing paradigms: A complete survey. *J. Syst. Archit.* **2019**, *98*, 289–330.
14. Halim, S.M.; Khan, L.; Thuraisingham, B. Next-location prediction using federated learning on a blockchain. In Proceedings of the 2020 IEEE Second International Conference on Cognitive Machine Intelligence (CogMI), Atlanta, GA, USA, 28–31 October 2020; pp. 244–250.
15. Bhagoji, A.N.; Chakraborty, S.; Mittal, P.; Calo, S. Analyzing Federated Learning Through an Adversarial Lens. In Proceedings of the International Conference on Machine Learning, Long Beach, CA, USA, 10–15 June 2019.
16. Multi-Access Edge Computing (MEC). Available online: https://www.etsi.org/technologies/multi-access-Edge-computing (accessed on 7 April 2022).
17. Mothukuri, V.; Parizi, R.M.; Pouriyeh, S.; Huang, Y.; Dehghantanha, A.; Srivastava, G. A Survey on Security and Privacy of Federated Learning. *Future Gener. Comput. Syst.* **2021**, *115*, 619–640.
18. Pop, C.D.; Antal, M.; Cioara, T.; Anghel, I.; Salomie, I. Blockchain and Demand Response: Zero-Knowledge Proofs for Energy Transactions Privacy. *Sensors* **2020**, *20*, 5678. [CrossRef]
19. Bryant, C.; Carvalho, W.; Baracaldo, N.; Ludwig, H.; Edwards, B.; Lee, T.; Molloy, I.; Srivastav, B. Detecting Backdoor Attacks on Deep Neural Networks by Activation Clustering. *arXiv* **2018**, arXiv:1811.03728.
20. Yao, Y.; Li, H.; Zheng, H.; Zhao, B.Y. Latent Backdoor Attacks on Deep Neural Networks. In Proceedings of the ACM SIGSAC Conference on Computer and Communications Security, London, UK, 11–15 November 2019.
21. Bolun, W.; Yao, Y.; Shan, S.; Li, H.; Viswanath, B.; Zheng, H.; Zhao, B.Y. Neural Cleanse: Identifying and Mitigating Backdoor Attacks in Neural Networks. In Proceedings of the IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 20–22 May 2019.
22. Wang, H.; Sreenivasan, K.; Rajput, S.; Vishwakarma, H.; Agarwal, S.; Sohn, J.; Lee, K.; Papailiopoulos, D. Attack of the Tails: Yes, You Really Can Backdoor Federated Learning. In Proceedings of the Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, Virtual, 6–12 December 2020.

23. Luca, M.; Song, C.; de Cristofaro, E.; Shmatikov, V. Exploiting Unintended Feature Leakage in Collaborative Learning. In Proceedings of the IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 20–22 May 2019.

24. Zhao, Y.; Chen, J.; Wu, D.; Teng, J.; Yu, S. Multi-Task Network Anomaly Detection Using Federated Learning. In Proceedings of the Tenth International Symposium on Information and Communication Technology, Hanoi-Halong Bay, Vietnam, 4–6 December 2019.

25. Nagar, A. Privacy-Preserving Blockchain Based Federated Learning with Differential Data Sharing. *arXiv* **2019**, arXiv:1912.04859.

26. Fereidooni, H.; Marchal, S.; Miettinen, M.; Mirhoseini, A.; Möllering, H.; Nguyen, T.D.; Rieger, P.; Sadeghi, A.; Schneider, T.; Yalame, H.; et al. SAFE Learn: Secure Aggregation for Private Federated Learning. In Proceedings of the IEEE Security and Privacy Workshops (SPW), San Francisco, CA, USA, 27 May 2021.

27. Liu, S.; Yu, G.; Yin, R.; Yuan, J. Adaptive Network Pruning for Wireless Federated Learning. *IEEE Wirel. Commun. Lett.* **2021**, *10*, 1572–1576. [CrossRef]

28. Chen, Z.; Tian, P.; Liao, W.; Yu, W. Zero knowledge clustering based adversarial mitigation in heterogeneous federated learning. *EEE Trans. Netw. Sci. Eng.* **2020**, *8*, 1070–1083. [CrossRef]

29. Shah, D.; Dube, P.; Chakraborty, S.; Verma, A. Adversarial Training in Communication Constrained Federated Learning. *arXiv* **2021**, arXiv:2103.01319.

30. Jeong, E.; Oh, S.; Kim, H.; Park, J.; Bennis, M.; Kim, S. Communication-Efficient On-Device Machine Learning: Federated Distillation and Augmentation Under Non-Iid Private Data. *arXiv* **2018**, arXiv:1811.11479.

31. Choudhury, O.; Gkoulalas-Divanis, A.; Salonidis, T.; Sylla, I.; Park, Y.; Hsu, G.; Das, A. Differential Privacy-Enabled Federated Learning for Sensitive Health Data. *arXiv* **2019**, arXiv:1910.02578.

32. Wang, X.; Garg, S.; Lin, H.; Hu, J.; Kaddoum, G.; Piran, M.J.; Hossain, M.S. Towards accurate anomaly detection in industrial internet-of-things using hierarchical federated learning. *IEEE Internet Things J.* **2021**, *9*, 7110–7119. [CrossRef]

33. Li, S.; Cheng, Y.; Liu, Y.; Wang, W.; Chen, T. Abnormal Client Behavior Detection in Federated Learning. *arXiv* **2019**, arXiv:1910.09933.

34. Gupta, R.; Kurtz, Z.T.; Scherer, S.; Smereka, J.M. Open Problems in Robotic Anomaly Detection. *arXiv* **2018**, arXiv:1809.03565.

35. Thudumu, S.; Branch, P.; Jin, J.; Singh, J. A comprehensive survey of anomaly detection techniques for high dimensional big data. *J. Big Data* **2020**, *7*, 1–30.

36. van Dijk, M.; Nguyen, N.V.; Nguyen, T.N.; Nguyen, L.M.; Tran-Dinh, Q.; Nguyen, P.H. Asynchronous Federated Learning With Reduced Number of Rounds and With Differential Privacy From Less Aggregated Gaussian Noise. *arXiv* **2020**, arXiv:2007.09208.

37. Hu, R.; Guo, Y.; Li, H.; Pei, Q.; Gong, Y. Personalized Federated Learning with Differential Privacy. *IEEE Internet Things J.* **2020**, *10*, 9530–9539. [CrossRef]

38. Bibikar, S.; Vikalo, H.; Wang, Z.; Chen, X. Federated Dynamic Sparse Training: Computing Less, Communicating Less, Yet Learning Better. *arXiv* **2021**, arXiv:2112.09824.

39. Jiang, Y.; Wang, S.; Valls, V.; Ko, B.J.; Lee, W.H.; Leung, K.K.; Tassiulas, L. Model Pruning Enables Efficient Federated Learning on Edge Devices. *arXiv* **2019**, arXiv:1909.12326. [CrossRef]

40. Truex, S.; Baracaldo, N.; Anwar, A.; Steinke, T.; Ludwig, H.; Zhang, R.; Zhou, Y. A Hybrid Approach to Privacy-Preserving Federated Learning. In Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security, London, UK, 15 November 2019.

41. Mahmood, Z.; Vacius, J. Privacy-Preserving Block-chain Framework Based on Ring Signatures (RSs) and Zero-Knowledge Proofs (ZKPs). In Proceedings of the International Conference on Innovation and Intelligence for Informatics, Computing and Technologies (3ICT), Sakheer, Bahrain, 20–21 December 2020.

42. Xie, T.; Zhang, J.; Zhang, Y.; Papamanthou, C.; Song, D. Libra: Succinct Zero-Knowledge Proofs with Optimal Prover Computation. In Proceedings of the Annual International Cryptology Conference, Santa Barbara, CA, USA, 18–22 August 2019.

43. Samangouei, P.; Kabkab, M.; Chellappa, R. Defense-GAN: Protecting Classifiers Against Adversarial Attacks Using Generative Models. *arXiv* **2018**, arXiv:1805.06605.

44. Li, D.; Wang, J. Fedmd: Heterogenous Federated Learning via Model Distillation. *arXiv* **2019**, arXiv:1805.06605.

45. GitHub—Zalandoresearch/Fashion-Mnist: A MNIST-Like Fashion Product Database. Benchmark. Available online: https://github.com/zalandoresearch/fashion-mnist (accessed on 23 September 2021).

46. Mahmood, Z.; Jusas, V. Implementation Framework for a Blockchain-Based Federated Learning Model for Classification Problems. *Symmetry* **2021**, *13*, 1116. [CrossRef]

47. Mikolajczyk, A.; Grochowski, M. Data Augmentation for Improving Deep Learning in Image Classification Problem. In Proceedings of the International Interdisciplinary PhD Workshop (IIPhDW), Swinoujscie, Poland, 9–12 May 2018.

48. Kherif, F.; Latypova, A. Principal Component Analysis. In *Machine Learning: Methods and Applications to Brain Disorders*; Mechelli, A., Vieira, S., Eds.; Elsevier: Amsterdam, The Netherlands, 2020; pp. 209–225.

49. Morsbach, F.J. Hardened Model Aggregation for Federated Learning backed by Distributed Trust Towards decentralizing Federated Learning using a Blockchain. Master's Thesis, Uppsala University, Uppsala, Sweden, 2020.

50. Kumar, R.; Khan, A.A.; Kumar, J.; Golilarz, N.A.; Zhang, S.; Ting, Y.; Zheng, C.; Wang, W. Blockchain-Federated Learning and Deep Learning Models for COVID-19 Detection Using CT Imaging. *IEEE Sens. J.* **2021**, *2007*, 06537. [CrossRef]

51. Rajalakshmi, K.; Tuhina, S. A Brief Analysis of Blockchain Algorithms and Its Challenges. In *Architectures and Frameworks for Developing and Applying Blockchain Technology*; IGI Global: Hershey, PA, USA, 2019. [CrossRef]

52. Sharma, D.K.; Pant, S.; Sharma, M.; Brahmachari, S. Cryptocurrency Mechanisms for Blockchains: Models, Characteristics, Challenges, and Applications. In *Handbook of Research on Blockchain Technology*; Elsevier: Amsterdam, The Netherlands, 2020; pp. 323–348.
53. Weng, J.; Weng, J.; Zhang, J.; Li, M.; Zhang, Y.; Luo, W. DeepChain: Auditable and Privacy-Preserving Deep Learning with Blockchain-Based Incentive. *IEEE Trans. Dependable Secur. Comput.* **2021**, *18*, 2438–2455. [CrossRef]
54. Kim, H.; Park, J.; Bennis, M.; Kim, S.L. Blockchained on-device federated learning. *IEEE Commun. Lett.* **2019**, *24*, 1279–1283. [CrossRef]
55. Maymounkov, P.; Mazieres, D. Kademlia: A peer-to-peer information system based on the xor metric. In *Peer-to-Peer Systems. IPTPS 2002*; Druschel, P., Kaashoek, F., Rowstron, A., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2002; Volume 2429.