SHERPA: Explainable Robust Algorithms for Privacy-Preserved Federated Learning in Future Networks to Defend Against Data Poisoning Attacks

Chamara Sandeepa*, Bartlomiej Siniarski†, Shen Wang‡, Madhusanka Liyanage§
*†‡§*School of Computer Science, University College Dublin, Ireland*
**abeysinghe.sandeepa@ucdconnect.ie,* †*bartlomiej.siniarski@ucd.ie,* ‡*shen.wang@ucd.ie,* §*madhusanka@ucd.ie*

***Abstract*—With the rapid progression of communication and localisation of big data over billions of devices, distributed Machine Learning (ML) techniques are emerging to cater for the development of Artificial Intelligence (AI)-based services in a distributed manner. Federated Learning (FL) is such an innovative approach to achieve a privacy-preserved AI that facilitates ML model sharing and aggregation while keeping the participants' data at the original source. However, recent research has investigated threats from poisoning attacks in FL. Several robust algorithms based on techniques such as similarity metrics or anomaly filtering are proposed as solutions. Yet, these approaches do not focus on investigating the intentions of the attackers or providing justifications and evidence for suspecting the behaviour of clients who are considered poisoners. Therefore, we propose SHERPA, a robust algorithm that uses Shapley Additive Explanations (SHAP) to identify potential poisoners in an FL system. Based on this, we develop a novel algorithm to differentiate poisoners via feature attribution clustering. We launch data poisoning attacks for different scenarios on multiple datasets and showcase our solution to mitigate the attacks. Furthermore, we show that privacy-targeted poisoning attacks can be mitigated with our approach. Accompanying the Explainable AI (XAI) technique for defence, our study reveals the potential for post-hoc feature attributions in countering data poisoning attacks with better explainability and improved justification in eliminating potentially malicious clients in the aggregation process.**

***Index Terms*—Federated Learning, XAI, SHAP, Data Poisoning, Beyond 5G/6G, Privacy, Defence, Robust Algorithm**

1. Introduction

The advent of FL [1] has brought forth a transformative approach to collaborative ML, allowing models to be trained across a distributed network of devices while respecting the privacy of data contributors. In FL, instead of directly transmitting data to a central server, ML models are trained locally by client devices. These models will then be forwarded to the server that aggregates them to create a more generalised global model. Thus, data will be kept locally within the client device without being accessed by a third party. This paradigm holds great promise in the context of future AI-based services, spanning diverse domains such as healthcare, zero-touch networks, finance, and the Internet of Things (IoT), where the need for collective intelligence without data centralisation is paramount.

Third-party service providers can utilise FL for training models in a distributed manner from millions of client devices using their local and private data without requiring them to share data directly. Furthermore, next-generation network architectures are envisioned to be fully AI-driven [2], [3]. Therefore, they can be expected to rely highly on up-to-date ML models, continuously trained by a distributed and privacy-preserved architecture like FL. However, a surge in identifying new threats and vulnerabilities in the privacy of FL has been observed over recent years, which raises the important requirement of solving these issues before fully adopting the technique in associated service applications. Examples of such attacks include property inference [4], [5], [6], membership inference [7], model inversion [8], [9], and poisoning [6], [10], [11]. Among them, inference and model inversion attacks mainly target invading privacy.

Unlike these common privacy-related attacks, poisoning attacks are widely known to be a security-related threat rather than privacy since they generally aim to manipulate model updates to make inaccurate predictions and damage the model's integrity. However, several recent works show poisoning can be used as a tool in FL to infringe privacy via these manipulations in the models [6], [7], [12]. Here, data poisoning aims to introduce backdoors, triggers, or biases in the final models. Privacy can get compromised when the attacker gets information on a particular target data when that data corrects the decision boundary of a biased model during training. Then, the attacker can identify that the particular target data is now included in the training process. Alternatively, a trigger will get activated when the model is trained by target data with a property the attacker is interested in. If this property is private and sensitive to individuals, their privacy will be affected significantly.

Therefore, if poisoning is detected early on and the elimination of poisoning clients is possible, potential problems in data leakage will be eliminated. Several methods are available for detecting poisoners, which use techniques like model similarity metrics [13], [14], [15], [16] and history-based reputation scores [14]. However, if a new client is added, the possibility of determining such clients without having a previous history is challenging. Further, the similarity metrics may not fully cover a proper explanation

for eliminating a client; instead, they are only focused on the majority and removing outliers. Therefore, when the majority is malicious, these methods often may not work properly. If there is a sufficient explanation when detecting a potential malicious client, proper reasoning and justification will be available to remove them. Thus, explainability would be a critical requirement since the decisions taken against FL poisoning should be understood by humans if needed, and proper evidence and justifications should be provided for necessary actions taken against the poisoners over the FL network. Therefore, a robust system that provides an interpretable solution for FL is required. For this, we propose a novel mechanism for detecting poisoners to assess the quality of local model updates in an FL system and eliminate potential poisoners in a justifiable manner.

1.1. Our Contributions

To the best of our knowledge, this is the first work to use the clustering of SHAP feature attributions to design a robust algorithm in FL to detect and eliminate poisoning influence from the system. We further highlight our key contributions as follows:

- We propose *SHERPA*, a novel, explainable, robust algorithm designed to detect and eliminate poisoners from an FL system. We use SHAP feature attributions to identify class-based key feature importances of a client that are contributing to making it a poisoner. Our solution aims to provide better justification for excluding and penalising any suspicious client by considering their behaviour via anomalies in feature attributions using HDBSCAN clustering. With our approach, the aggregator can explain the reason for any elimination since deviations in feature attributions can demonstrate the model is not prioritising important features for the target classes. We identify that no comprehensive investigation is done on using post-hoc feature attribution techniques like SHAP as a potential defence mechanism. Therefore, with our approach, we bring the requirement of improving the explainability in detecting the poisoners.
- We show that our proposed solution can detect poisoned clients even when 80% of the clients are poisoned with up to 98% detection accuracy. The experimental results show that the proposed solution can perform better than the state-of-the-art algorithms like [13], [14], [17] in detecting poisoned clients. We also compare our approach with varying parameters and multiple datasets to further analyse the behaviour of the solution under different configurations and scenarios.
- Our work discusses and demonstrates how poisoning can impact the privacy of clients in a practical use case and showcases the possibility of mitigating the issues by eliminating the poisoners.

1.2. Outline

The rest of the paper is arranged as follows: Section 2 discusses related work associated with poisoning attacks on FL. Section 3 provides an overview of the problem statement with the system and threat models. Section 4 discusses the proposed FL robust defence framework architecture along with the novel defence algorithm. Section 5 provides experiments performed on the algorithm. Discussions on results are provided in Section 6. Finally, the paper concludes along with future works in Section 7.

2. Related Works

This section provides a brief overview on the FL-based poisoning attacks and its state-of-the art defence mechanisms.

2.1. Poisoning Attacks on FL

In FL, the models that clients forward can potentially leak information on the data and its properties. Based on exploiting the model parameters shared, several attacks have been identified. Common types of attacks on FL include membership inference [7], [18], which attempts to infer membership state of a client regarding the participation in the FL process, property inference [5], [6], that aims to detect specific properties or features in the dataset not relevant to the main task, data reconstruction attacks [9] that attempts to recover original dataset or part of it via techniques like reversing the gradients shared by clients. These attacks can be considered passive since they primarily launch the attack by evaluating the received model updates. However, the practicality of the attacks like inference may lie in the requirement of correctly identifying the decision boundary changes on the models with each update [6], [19], which could be difficult unless the attacker has a highly accurate understanding of the model behaviours.

For this, the attackers incorporate poisoning attacks to artificially alter the decision boundary of FL models [6]. Therefore, poisoning is a major issue that comes as a significant threat to both the security and privacy of FL. These attacks then be a boosting technique for inference attacks to improve their attack success rate [6], [7]. Further, attackers also aim to compromise the utility of the models via poisoning updates. Backdoors and trigger attacks from poisoning can also affect the utility of the model for a targeted set of classes. These triggers are also used for privacy leakages, where the trigger is activated when a specific property or data in the private dataset appears in a target client [6], [20]. Therefore, for practically implementing decentralised FL applications in future B5G/6G networks, early detection of poisoning and elimination are essential requirements to be addressed.

2.2. Existing Robust Algorithms

Several robust algorithms that aim to mitigate poisoning attacks on FL are introduced in the research literature. The

following are some of the key existing techniques used for detection and elimination of clients:

Krum [13]: This method considers the similarity between the participant's updates by assuming a poisoner would propose an arbitrary gradient update compared with a benign client. However, for better accurate results, this method requires an estimation of the number of poisoners in the network, which is unlikely to be determined early.

FoolsGold [14]: This technique assigns cosine similarity-based reputation scores to participants based on their historical contributions and uses these scores to weigh the influence of their updates during aggregation. It may not perform well if these reputation scores are not available. Thus, this method may not recognise poisoners that appear dynamically.

Trimmed Mean [21]: The trimmed mean is an aggregation rule where the server identifies $k; k < n/2$ trim parameters and eliminates the largest and smallest k values while aggregating the remaining $n - 2k$ values. This means that the maximum number of malicious clients should be less than 50% of total clients.

Median [21]: In this technique, the server considers the median value of each parameter received from clients to minimise the effect of poisoners. This also has the issue of assuming the system has less than 50% for malicious clients.

FLTrust [15]: This mechanism is also an aggregation rule which uses ReLU-clipped cosine similarity-based trust scores to aggregate model updates. They use a root dataset, maintain a separate model in the server, and assume the root dataset is clean and generally represents the overall client models. Therefore, this method may not work if client model data deviates from the root dataset or the root dataset itself is poisoned if taken from a third party. In our work, we also use representative samples for deriving feature attributions. However, they can be random values or zero vectors if a trustworthy dataset is not available.

FLAME [16]: Here, the authors use a combined cosine distance and clustering with Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN) for determining poisoners. However, they assume a majority of the clients (>50%) are benign, which can be a limiting factor with these distance-based metrics. Furthermore, they inject noise into the model updates, which can also affect the model performance and utility.

MOAT [17]: This work uses SHAP feature attributions for assessing poisoning in FL. However, they do not consider a clustering approach for poisoning detection and use a z-score of these features and a dynamic hyperparameter ϵ value that is set as a threshold for anomaly detection. This value can be difficult to estimate early, and it can vary significantly depending on the nature of the dataset. Our approach does not rely on an arbitrary single value and uses the combined behaviour of feature attributions itself to determine the poisoners.

The majority of these techniques use general model similarity metrics and do not consider the factors of how these mechanisms work. Furthermore, some of them tend to use arbitrary values assigned in the algorithms that do not have clear justifications for why the mechanism works with these values. We consider these limitations and propose a robust aggregation mechanism with SHAP that aims to clarify the reasons for eliminating any suspicious client who can be a potential adversary.

3. Problem Statement

3.1. System Model

The basic components and actors of a FL-based system include the following:

- **Service provider** - This entity is responsible for identifying a service requirement and delivering the design and infrastructure to facilitate the ML-based service. The service could either be a third-party application subscribed by a user or a background utility-based service application.
- **User device** - The user subscribes to a service via a user device. The users generate and store data in the end-user device where the local models are trained.
- **Aggregator** - The aggregator is performing the averaging of all the models received from clients. The aggregation server can be the cloud server itself, or it may also be an edge server close to the user device. This is to reduce any latencies occurring in communication between clients and servers during model transmission rounds. Upon training, the updated global model is forwarded back to the main service provider.

An overview of the system model components and their interactions are shown in Fig. 1.

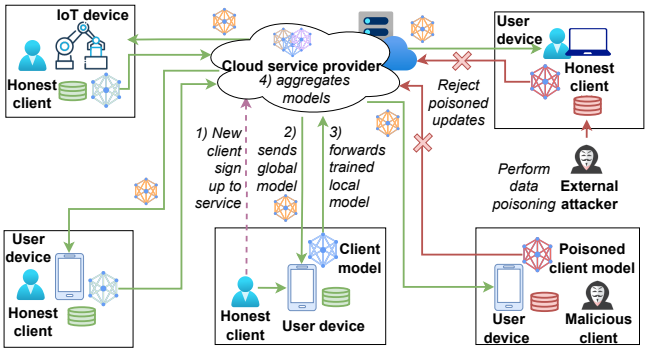


Figure 1: FL system model with multiple client actors

3.2. Threat Model

An adversary may attempt to poison the data in the user device to make the local model biased or backdoored. Suppose these poisoned models are aggregated by the server. In that case, the attacker may be able to take control of the model training process and also evade the privacy of individuals via backdoor and inference attacks.

3.2.1. Poisoning Attacks. Here, numerous poisoning attacks can be launched by an attacker, all aiming to perturb the original dataset. The objective of the attacker could be to take control of the aggregated gradient movement towards a malicious objective to compromise the model utility. This can be done by randomly moving the gradient away from the optimal path or navigating the gradients of the global model to a shifted or biased gradient, which can lead the predictions to cause higher error towards a targeted set of classes, creating a backdoor. The aggregation of N gradients obtained via Stochastic Gradient Descent (SGD) in FL can be represented as:

$$F(\Delta W_1, ..., \Delta W_N) = \sum_{i=1}^{N} \lambda_i \Delta W_i \quad ; \forall i \lambda_i \neq 0 \qquad (1)$$

where λ_i is the scaling value assigned to each gradient ΔW_i. A set of k coordinated poisoning clients where a single poisoning client j having the gradient $\Delta\Psi_j$ can shift the overall model gradients towards a malicious objective gradient $\Delta\Psi$ if the combined poisoned gradients be:

$$\sum_{j=1}^{k} \lambda_j \Delta\Psi_j = \Delta\Psi - \sum_{i=k+1}^{N} \lambda_i \Delta W_j$$
$$\Rightarrow F(\Delta\Psi_1, ...\Delta\Psi_k, \Delta W_{k+1}, ...\Delta W_N) = \Delta\Psi \qquad (2)$$

Therefore, the attackers can poison their models to reach the target objective $\Delta\Psi$ by estimating the scaling factors and individual gradients based on the global model parameters from the previous round. This could either be done by manipulating the model parameters directly or poisoning the local data used to train the model. Local data poisoning can be considered a more straightforward approach when the malicious objective is focused towards a target set of output classes since modified training data can modify the local model updated towards the adversary goal.

We considered two types of data poisoning attacks that are based on output labels: 1) random label poisoning and 2) target label poisoning. For a poisoning client with dataset (X, Y) having N data samples, where $X = \{x_1, x_2, ..., x_N\}$ and $Y = \{y_1, y_2, ..., y_N\}$. Each value in Y is a label of any value within the set of possible class labels $C = \{1, 2, ..., c\}$. Let $S \subset \{m, ..., n\}$, which is the selected subset of indices of data points selected by the client for poisoning. In random poisoning, for each position of y value in the indices of S, the attacker modifies the label l of Y to any random label $\tilde{l} \in C$. In the case of target label poisoning, the poison label $\tilde{l}$ is a single or targeted subset of labels in C.

3.2.2. Privacy Attacks via Poisoning. The privacy of benign participants in an FL system is impacted if the attacker gains knowledge about the membership state of a particular user or the presence of a target property in the benign client's data. For this, the attacker typically uses a separate model called *attack model* Ω, which is trained to detect alterations in the decision boundary in the target features. This can be denoted as [6]:

$$(\Omega(f(M_t)) \geq \lambda)?P : \bar{P} \qquad (3)$$

Where the attack model Ω is provided with the target $f(M_t)$ at FL round t, which can either be the original model parameters of the model M_t or a set of predictions from M_t. Based on this input, Ω provides the probability of a target property or membership state denoted in P or not, shown by $\bar{P}$, thresholded by λ.

To train Ω, the attacker may use a set of *shadow models* $G = \{g_1, g_2, .., g_m\}$ that is similar in architecture to the target FL global model M. The shadow models are trained with attacker-generated data [7], [18], and the attacker obtains a dataset containing either the shadow model parameters or the shadow model prediction vectors with and without the desired property. This dataset is used to train Ω, and the attacker aims to detect the desired property changes in the target model M.

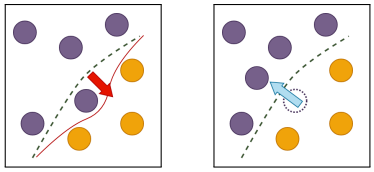


Figure 2: An adversary can poison the global FL model to make the decision boundary deviate from its original position. When benign clients correct that boundary, the adversary will know the target feature has appeared.

However, in reality, it may be difficult to determine the decision boundary via shadow model training alone. This could be because the shadow model may not exactly represent a typical FL scenario since they are isolated ML models. Furthermore, the attacker's dataset may not fully account for the general distribution of data in the client models. Therefore, to amplify the decision boundary changes from benign clients, the attacker uses poisoning attacks as a strategy [6]. Fig. 2 shows that the decision boundary is altered with poisoning, and when benign clients reset the boundary, the attacker can identify the changes. The attacker may alter the decision boundary related to a certain property P by first poisoning only the data with the target property and training the local model with the poisoned dataset. However, the attacker does not alter any data that does not have the property. This will result in a backdoor where the aggregated model will perform poorly only for the target property P. If an honest client trains their dataset with the target property, they will tend to correct the decision boundary of the global model. Once this change is detected by the attack model Ω, the attacker can conclude that the data with the target property P has appeared in the benign clients. If this property is specific to a particular individual, the attacker can also predict the membership state, verifying the participation of the target client in the FL process. This will result in a privacy breach if these

properties and membership states are considered private and sensitive attributes.

4. SHAP-based Defence

Future distributed ML services can be expected to adopt FL-based distributed architectures to achieve high privacy guarantees to the network's end users. However, a critical issue with such distributed, collaborative ML approaches is the ability of untrusted users to influence the learning process, unlike the current centralised data-based training. To eliminate the threat of poisoners in the system, this section proposes a custom architectural framework for FL-based future networks to implement defence strategies and eliminate threats from client models. To achieve enhanced explainability when realising the threat of poisoning attacks in FL learning systems, we develop a novel SHAP-based poisoning detection and defence strategy.

4.1. Defence System Architecture

Our proposed defence system involves multiple components, which primarily incorporate the followings:

- **FL aggregator** - The process of FL involves an aggregator, where we simulate a virtual aggregator that performs the Federated Averaging (FedAvg) aggregation function on the collected updates from clients.
- **Clients** - The clients receive initial model updates from the aggregator. Each client consists of a dataset, which is used to train the local model and the models are sent back to the aggregator.
- **FL Process configurations** - In our simulations, we design our system to configure client and aggregator for varying types of FL models and aggregation. The datasets are also configurable with different types and multiple strategies for poisoning, including random and targeted data poisoning. It can also configure the order and intervals of poisoning of clients.
- **Robust aggregation algorithm** - The algorithm is designed with SHAP, where the feature attributions are first obtained and clustered to detect poisoners. Then, the elimination of these potential poisoners is done such that the aggregation will occur by excluding the detected poisoning group.
- **Fine tuning and visualisation** - This component provides visualisation of feature attribution groups after clustering. It also provides options for hyper-parameter tuning in the robust algorithm.

Fig. 3 presents a high-level design of the attack and defence simulation system we developed.

4.2. Poisoning Detection via SHAP-based Feature Attribution: Overview of the Approach

In FL-based robust aggregation algorithms, we mainly observe that many defence mechanisms [13], [14], [15], [16]

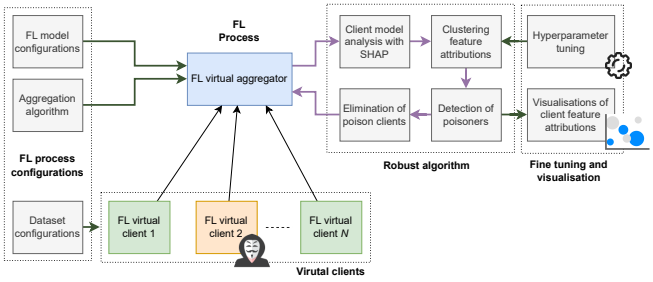


Figure 3: Overview of the defence system components

apply aggregation rules by evaluating the model parameters directly via techniques like cosine similarity. However, there are two main issues here:

4.2.1. Variation in Model Parameters. We identify that a possibility exists for model parameters to vary significantly from each other. For example, in the case of Neural Network (NN) models, model parameters can vary significantly from one layer to another, yet their predictions may have the same value. This makes distance metrics produce reduced similarities and misclassify benign clients as poisoners and vice versa. We show that the cosine similarity of two models M and $\tilde{M}$ can be different at a layer i when $\tilde{W}^{(i)} = \alpha W^{(i)}$, where α is the ratio between the outputs from $i-1$ layers of the two models, such that their predictions $Y = \tilde{Y}$ (more information on the derivation is available in Appendix B.1). Therefore, we observe that two models can have more variability in model parameters, yet their predictions can have similar results.

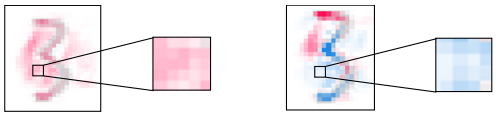
4.2.2. High Similarity in Poisoning Model Parameters to Benign Models. The next issue is that poisoners can manipulate the model parameters to craft an update similar to a benign client update. However, their output will be poisoned. For example, an attacker can modify a selected layer of a model $\tilde{M}$ like the output layer L, such that all other $L-1$ layers remain the same as a benign model M. However, this modification can result in a poisoned output $\tilde{Y}$, which is significantly different from the benign model output Y (information on the steps are provided in Appendix B.2). Hence, even when a model is poisoned, poisoning updates may not be detected if the similarity of the updates is high.

Therefore, instead of relying on the black-box model parameters, we propose the assessment of SHAP-based model output feature attributions, which can provide insights into the differences between the model behaviours, which is the important factor for both the poisoners and the real clients (more information on SHAP available in Appendix A.2). Despite how the black-box model parameters are arranged, we consider the honesty of an update to depend on the nature of the output they provide.

Even though model outputs can directly provide some level of information on the poisoning, many model predictions may be required to detect a poisoning scenario since the information from a prediction is limited to a

single prediction vector. Also, the amount of data available to obtain predictions for model analysis is limited on the aggregator side. Furthermore, developing another black-box ML model to predict the poisoning via model predictions would not justify the elimination of the poisoners due to lack of explainability.

4.2.3. Observation of Poisoners via SHAP. To mitigate the drawbacks of direct model outputs and enhance explainability, we use XAI-technique SHAP to design a new robust algorithm. Post-hoc explainers like SHAP can provide more information on the trained model beyond model predictions, which can inherently provide details on how well the model recognises and prioritises the features in input data that are relevant to decision-making. We make use of this property as the basis for our robust algorithm development. Suppose a benign local FL model is trained for handwritten digit recognition with MNIST dataset [22]. It outputs SHAP feature attributions for the output label 3 as in Fig.4a. Here, the high-intensity regions indicate high importance, and the red colour shows a positive value. Thus, high positive importance is given to the benign client, while a poisoned client, indicated in Fig. 4b, does not have a similar score in the same region.



(a) Benign (b) Poisoned

Figure 4: Comparison of feature attributions in class 3 from a benign client and a poisoned client. The benign client has a high feature importance score, marked in red; meanwhile, a poisoned client may not have similar importance for the same region.

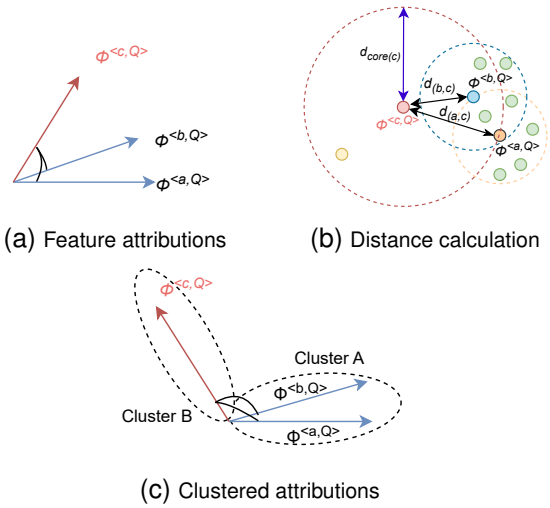
As shown by the comparison, the values of the features between the two clients can vary significantly in certain regions, which makes it easier to differentiate a poison client from a group of benign clients.

4.2.4. Clustering of Feature Attributions. The SHAP values for each client can be considered as a set of high-dimensional vectors. Suppose the SHAP feature attributions for 3 clients for the same class Q are given as $[\phi^{<a,Q>}, \phi^{<b,Q>}, \phi^{<c,Q>}]$. A simplified representation of these vectors mapped to a two-dimensional plane is shown in Fig. 5a. The cosine similarity between the two feature attributions of client a and b can be denoted as:

$$\cos(\mathbf{a}, \mathbf{b}) = \frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\| \cdot \|\mathbf{b}\|} \tag{4}$$

where $\|\mathbf{a}\|$ and $\|\mathbf{b}\|$ are the L2 norms of a and b.

However, in a case where the client c is malicious and placed close to a and b, a comparison of cosine similarity of c relative to a and b could indicate c as a benign client. Therefore, we use the HDBSCAN algorithm to account for the K-Nearest Neighbours (KNN) around each feature attribution, such that the differences are further enhanced by considering the similarity among nearby clients (more details in Appendix A.3).



(a) Feature attributions (b) Distance calculation

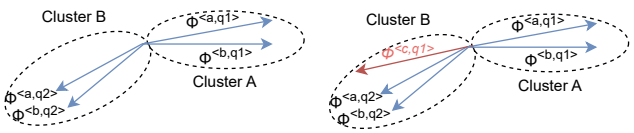
(c) Clustered attributions

Figure 5: Feature attribution distance calculation via HDBSCAN to increase the distance between suspicious attributions and a benign attribution cluster.

Fig. 5b shows an example of HDBSCAN operation of accounting to both Euclidean distances and KNN core distances among the clients a, b and c. Here, a sample of 5 neighbours is taken to set the core distance. The mutual reachability distance d_{mreach} is the metric used in the HDBSCAN algorithm when developing clusters. Given two points, the d_{mreach} takes the maximum distance of $(d_{core(a)}, d_{core(b)}, d_{(b,c)})$ between the two points. In clients a and c, the Euclidean distance $d_{(a,c)}$ equals the core distance of c. Therefore, d_{mreach} for a and c can be set as $d_{(a,c)}$. However, for b and c, $d_{core(c)} > d_{(b,c)}$, which makes $d_{mreach} = d_{core(c)}$. When obtaining the d_{mreach} between clients, they should either remain similar to the Euclidean distance or be increased. Then, the cosine similarity between clients b and c with d_{mreach} should also be reduced with increasing distance (described in Appendix C) since $d_{mreach(b,c)} > d_{(b,c)}$. Hence, with low similarity, it is easier to distinguish c in a separate cluster as represented in Fig. 5c.

4.2.5. Detection of a Poisoning Scenario. The clustering of SHAP values includes the features of the same class assigned in a unique cluster. However, if a malicious client aims to poison a class q_2 to q_1, their model will result in output q_1 for the inputs corresponding to q_2. Fig. 6 represents a scenario where the SHAP values of a poisoned client are introduced to the clustered attribution map with benign clients. As the feature attributions vector of the poisoned client are clustered at B, their class label should be q_2. However, the SHAP values are producing this vector under the class q_1, which indicates a label poisoning from class

q_2 to q_1. Therefore, the aggregator can consider excluding this client during aggregation.



(a) Benign cluster (b) Poisoned cluster

Figure 6: Feature attributions for a class q_1 in the cluster with feature attributions similar to q_2 leads to contradiction and results in being recognised as a poisoning behaviour.

The next sections describe the detailed steps for implementing the poisoning detection and defence algorithm.

4.3. Poisoning Detection Algorithm

We propose a novel framework named SHap-based Explainable Robustness against Poisoning Algorithm (SHERPA) for detecting and defending against different types of data poisoning attacks done on FL local models.

4.3.1. Deriving Feature Attributions. The first step of the algorithm is to obtain a set of feature attributions from local models. A local model m_i with class labels $C = \{1, 2, ..., c\}$ is received by the aggregator. The SHAP feature attributions $\phi^{<i,j>}$ obtained for each output class j of client i can then be represented as $\phi^{<i,j>} = \{\phi_1, \phi_2, .., \phi_k\}$ for k set of features. These feature attributions require a sample value x_s to the SHAP as shown in Equation 10 (in Appendix A.2). The server can maintain a small sample dataset D_s where a random subset $d_s \subseteq D_s$ is used for the baseline input data for deriving the feature attributions from SHAP. During the initiation of the FL service, the service provider trains the first version of the global model with their local data. This local data can be used as the baseline dataset D_s for model evaluations. The baseline dataset can also be generated by generative AI techniques [23]. An alternative is to use zero vector [17] data if such sample data is not available. This baseline data will be used in common across all the clients to derive SHAP the feature attributions. Therefore, for C classes, the list of all feature attributions per client $i \in N$ in a FL system with $N = [1, n]$ clients at FL global round t can be denoted as $\phi_t^{<i>} = \{\phi_t^{<i,1>}, \phi_t^{<i,2>}, ..., \phi_t^{<i,c>}\}$. Thus,

$$\phi_t^{<i>} = \bigcup_{j=1}^{c} \phi_t^{<i,j>} \tag{5}$$

Therefore, for all N clients, the total set of features corresponding to sample data x_s in d_s can be represented as:

$$\gamma_{tx_s} = \begin{bmatrix} \phi_t^{<1>} \\ \vdots \\ \phi_t^{<n>} \end{bmatrix} = \begin{bmatrix} \phi_t^{<1,1>} & \dots & \phi_t^{<1,c>} \\ \vdots & \ddots & \vdots \\ \phi_t^{<n,1>} & \dots & \phi_t^{<n,c>} \end{bmatrix}$$

This matrix γ_{tx_s} can be used as the input for the clustering algorithm Θ_t, which can analyse the probability density distributions among the clients. For each x_s value in d_s, we obtain γ_t, such that we form the final δ_t list that consists of all values of feature sets corresponding to each element in the subset d_s with r values as $\delta_t = \{\gamma_{tx_1}, \gamma_{tx_2}, ..., \gamma_{tx_r}\}$. The steps are illustrated in Algorithm 1.

Algorithm 1 Deriving Feature Attribution Inputs

Input: Sample baseline input data value x_s with F features for each data point, ML model function f of client i, set of class labels C
Output: SHAP feature attributions for client i
function COMPUTESHAPVALUES()
  Let $\phi^{<i>} = \{\}$
  for Class j in C **do**
    Let $\phi^{<i,j>} = \{\}$
    for Feature l in F **do**
      $\phi_i(l,j) = \sum_{S \subseteq F \setminus \{l\}} \frac{|S|!(|F|-|S|-1)!}{|F|!} \{f(x_s \cup \{l\}) - f(x_s)\}$
      $\phi^{<i,j>} = \phi^{<i,j>} \cup \phi_i(l,j)$
    end for
    $\phi^{<i>} = \phi^{<i>} \cup \phi^{<i,j>}$
  end for
  Return $\phi^{<i>}$
end function
Input: List of client models $M = \{m_1, m_2, ..., m_n\}$ at round t, sample baseline data D_s, a set of class labels C of a model
Output: List of input feature attributions for clustering δ_t
function GETFEATUREATTRIBUTIONS()
  Let $d_s = \{x_1, x_2, ..., x_p\}; d_s \subseteq D_s$
  Let $\delta_t = \{\}$
  for x_s in d_s **do**
    Let $\gamma_{tx_s} = \{\}$
    for model m_i in M **do**
      $\phi_t^{<i>} =$ COMPUTESHAPVALUES(x_s, m_i, C)
      $\gamma_t = \gamma_{tx_s} \cup \phi_t^{<i>}$
    end for
    $\delta_t = \delta_t \cup \gamma_{tx_s}$
  end for
  Return δ_t
end function

4.3.2. Clustering of Attributions. The clustering mechanism implemented in our approach is HDBSCAN, where each feature of $\phi^{<i,j>}$ for all γ_{tx_s} in δ_t is assigned a cluster label θ as $\Theta_t = [\theta_{x_1}^{<1,1>}, ..., \theta_{x_1}^{<n,c>}, ..., \theta_{x_p}^{<1,1>}, ..., \theta_{x_p}^{<n,c>}]$. Here, two values of θ may have the same cluster label.

We consider the similarity of explanations of different clients. Feature attributions belonging to the same class, even if the clients are different, should have high probability density among each other, such that the mutual reachability distance among similar feature attributions is low and should be grouped in the same cluster. Therefore, when considering

two clients a and b, if any one of the clients is not poisoned, their SHAP feature attributions for the same class q should be included in the same cluster. This can be denoted as:

$$\theta_{x_s}^{<a,q>} = \theta_{x_s}^{<b,q>} \quad ; q \in [1, c], x_s \in |d_s| \tag{6}$$

for two feature attributions a and b. However, if a client is poisoned, feature attributions of the poisoned elements tend to deviate from the same feature attributions, forming either an outlier or belonging to a different cluster than its same feature attribution cluster. Thus, in such cases, the above equality among two attributions a and b does not hold, making $\theta_{x_s}^{<a,q>} \neq \theta_{x_s}^{<b,q>}$. This property is used to detect the poisoners from benign clients.

4.3.3. Implementing Poisoning Client Detection via Search by Cluster. For the search for poisoners in implementation, it is easier to look from the perspective of a cluster instead of comparing each client. Suppose the unique values for cluster labels $u(\Theta)$ of Θ_t are given as $u(\Theta) = \{\theta_i | \theta_i \in \Theta\}_{i \in \{1,..,|\Theta|\}}$. For each unique cluster label, the above relationships can be evaluated among different clients who belong to that cluster. For each cluster, we iteratively search for clients who does not satisfy the condition in Equation 6. Algorithms 2 and 3 provides the the detailed process of detecting the poisoned clients.

Algorithm 2 Get Significant Client Label for Cluster

Input: set of client models W_θ available in a cluster θ_i, list of client class labels Q_i shown by each client in cluster θ_i, sample baseline data D_s
Output: Class label q_c considered in the cluster θ_i
function GETCLIENTLABELFORCLUSTER()
  if $q_i = q_j; \forall q_i, q_j$ in Q_i **then** ▷ Same client label
    $q_c = q_i$
  else
    $q_c = -1$ ▷ Cluster has conflicting labels, considering malicious
  end if
  Return q_c
end function

4.4. Defending Against Poisoning

To defend the FL system against poisoning, the feature attributions that do not belong to the same class should be eliminated or suppressed before the aggregation. For this, the aggregator maintains a record of suspicious score S_t^k for each client k at round t. If a certain client exceeds a suspicious score T_h, then it will be eliminated. Here, we define $T_h = \frac{n_{S_t}}{n}$, where n is the total number of clients and $n_{S_t} = \sum_{i=1}^{n} S_t[i]$. Thus, T_h denotes the average suspicious score per client. The clients who exceed the threshold average suspicious score will be eliminated. For the other clients, we consider the suspicious score as a down-scaling factor, which will be used before the aggregation as:

$$w_t \leftarrow \sum_{k=1}^{K} \left(1 - \frac{S_i^k}{|C||d_s|}\right) \frac{n_k}{n_t} w_t^k \tag{7}$$

where the global model parameter w_t at round t is obtained by getting the sum of all parameters of K clients from which a client k has n_k data samples out of n_t total samples, and C is the set of classes, while d_s is the sample baseline subset used to derive the SHAP values.

The elimination process is presented in Algorithm 3.

Algorithm 3 Elimination of Suspicious Clients

Input: The set of normalised client weights $W = \{\tilde{w}_t^k; k \in [1, n]\}$, Feature attributions list δ_t for round t, threshold score T_h for poisoning clients, suspicious counts list from previous round for each client $S = \cup_{k=1}^{n} s_{t-1}^k$, probability density-based clustering algorithm function Θ
Output: Aggregated weights w_t for round t
function ELIMINATESUSPICIOUSCLIENTS()
  Get $\Theta(\delta_t) = \bigcup_{p=1,q=1,x_s=1}^{n,c,|d_s|} \theta_{x_s}^{\langle p,q \rangle}$ ▷ Clustering
  Let $u(\Theta) = \{\theta_i | \theta_i \in \Theta\}_{i \in \{1,..,|\Theta|\}}$ ▷ Unique IDs
  Let $S_t = \mathbf{0}_n$ ▷ Suspicious scores list for each client at current round t
  for each cluster ID $i \in u(\Theta)$ **do**
    $\theta_i = \cup\{\theta_{x_s}^{<p,q>} \mid \theta_{x_s}^{<p,q>} \in \Theta, |\theta_{x_s}^{<p,q>}| = i\}$
    $W_\theta = \cup\{W[p] \mid \theta_{x_s}^{<p,q>} \in \theta_i\}$▷ client models in cluster θ_i
    $Q_\theta = \cup\{q \mid \theta_{x_s}^{<p,q>} \in \theta_i\}$ ▷ client labels in θ_i
    $q_c =$ GETCLIENTLABELFORCLUSTER(W_θ, Q_θ)
    for each $\theta_{x_s}^{\langle p,q \rangle}$ **do**
      $$S_t[p] = S_t[p] + \begin{cases} 1 & \text{if } q \neq q_c \\ 0 & \text{otherwise} \end{cases}$$
    end for
  end for
  Let $\psi = \{\}$
  for client p in S_t **do**
    if $S_t[p] \geq T_h$ **then** $\psi = \psi \cup 0$
    else $\psi = \psi \cup \left(1 - \frac{S_i^p}{|C||d_s|}\right)$
    end if
  end for
  $S = S + S_t$
  Return $w_t \leftarrow \sum_{k=1}^{K} \left(\psi[k].\tilde{w}_t^k\right)$
end function

5. Experiments

This section outlines and discusses the experimental findings on the algorithm and a comparison of it with other related works. We also apply the attack to a use case of facial recognition where we simulate a privacy-related PAPI attack from [6] and show the attack is mitigated with the proposed approach.

5.1. Experimental Setup

The experiments were performed on a simulated FL environment where we implemented the defence system

components with Pytorch and Flower [24] frameworks. The system is run in a server instance with NVIDIA A4000 GPU, Intel Xeon 2.10 GHz CPU with 20 cores, and 128 GB RAM. We used multiple datasets for our experiments together with their associated model configurations as follows:

5.1.1. MNIST. This is a set of 60,000 handwritten numbers from 0-9. Each data consists of 28x28 pixel grayscale images. We used a Convolutional Neural Network (CNN) model with 2 convolutional layers, where the first layer consists of 32 output channels with kernel size 3 and stride 1. The second convolutional layer outputs 64 channels with the same kernel size and stride of 3 and 1, respectively. It is followed by two dropout layers and two fully connected layers. The final output is given by a log softmax layer that predicts the number denoted in the input image.

5.1.2. Fashion. Fashion dataset [25] is a collection of 28x28 grayscale images of 70,000 fashion products from 10 classes, with 7,000 images for each class. For this dataset, we use a CNN model similar to MNIST with 2 convolution layers followed by dropout and fully connected layers.

5.1.3. NSL-KDD. This dataset consists of 125,973 data records and serves as a benchmark for Intrusion Detection Systems (IDS) encompassing diverse network attack scenarios. For our experiments, we specifically focus on two prominent categories of traffic: normal and Denial of Service (DoS) attacks. The model used for this dataset is a NN model with 512 nodes in a linear layer followed by dropout layer and another linear layer with output classes for the two categories.

5.1.4. 5G-NIDD. 5G-Network Intrusion Detection Dataset (5G-NIDD) [26] is a recent IDS dataset with 9 types of network attacks. This dataset consists of data extracted from DoS and port attack scenarios. The dataset consists of 52 attributes on the network properties and the attack status. In this dataset, we selected the two main categories, benign and UDPFlood attack, for our experiments. We used an NN model with three linear layers with 89, 30 and 2 nodes for the training with the dataset.

5.1.5. CelebA. CelebFaces Attributes Dataset (CelebA) dataset consists of 202,599 face images of the size 178×218 with 40 binary attribute annotations describing the properties of each face. We resize and normalise the input dataset to a 32x32 size for faster training and aggregation of client models. This model also consists of two convolutional layers with kernel size 5 each and a max pool layer with kernel size 2 and stride 2. This is followed by 3 linear layers, which outputs the target set of classes. Our use case here is to classify the property of two genders, either male or female, from the input images.

In the experiments, we first observe the effects of poisoning attacks shown by the SHAP feature attributions on the model predictions.

5.2. Visualisation of Poisoning: Two Scenarios

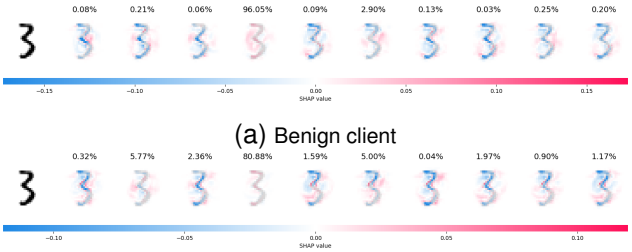
Here, we performed random and targeted poisoning on a system of 10 clients where 50% of the clients were poisoned. To illustrate the effect of poisoning, we use the MNIST dataset. For random poisoning, we modified all labels of the clients. In targeted poisoning, we set all the labels to the target class 3. The local models are trained for 10 FL rounds.

5.2.1. SHAP Feature Attributions Maps. We use Algorithm 1 and obtain the SHAP values for each client after training. For each client, the SHAP feature attribution mapping is obtained for a sample random input data. For all datasets, the feature attributions are equal to the number of features in the dataset, which are represented for each class. For example, in MNIST, each feature attribution map consists of a 28x28 set of values for each of the ten classes for the 0-9 digits. This is obtained for all the clients in the FL system. Fig. 7 shows sample feature attribution values mapped for a benign client and clients with random and targeted poisoning trained on MNIST.

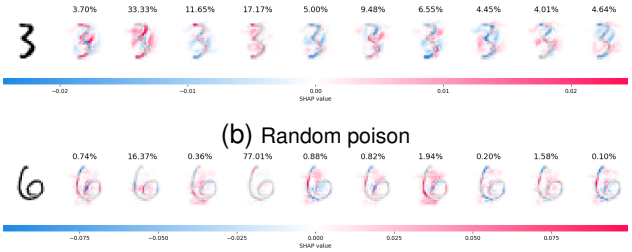
The feature attributions for all 10 classes given the input value 3 for a benign client are shown in Fig. 7a. We use the gradient explainer from SHAP to obtain the feature attributions. The probability values show that the client correctly identifies the expected class as 3; meanwhile, the feature attributions also show more importance for class 3 than the other classes. However, for a poisoned client with random poisoning, the attributions show feature importance for multiple random classes, while the model cannot correctly classify the input class 3. Therefore, the feature attributions do not show a clear result in random poisoning. Thus, we can consider that the decision boundaries for all classes have shifted, supporting the poisoner's objective to result in a utility drop when aggregating the poisoned model.

In the target poison, the attacker selects label 3 to be the target label such that the accuracy does not drop for that label. Therefore, the SHAP attributions also do not vary significantly from a benign client for label 3. However, the goal of the attacker is to modify the decision boundary of the model such that if any other input is selected, they are also classified as 3, as shown in Fig. 7d. Here, we observe the SHAP attributions behave similarly to a random poison scenario, which can be expected since the other class labels than the target class are swapped similarly to random poison.

5.2.2. Clustering of SHAP Feature Attributions. We then use the HDBSCAN clustering technique to obtain the relationship between each feature attribution set. The clustering is done for all feature attributions from the clients. The resulting clusters are mapped to t-SNE plots such that one single point in the plot represents the mapping of total feature attributions of a given client for a specific class. Fig. 8 presents these t-SNE projections of the clusters for a scenario of targeted poisoning of MNIST where two target labels, 1 and 7, are swapped in 3 out of 10 clients. Here, we mark each point by its corresponding class. From the



(a) Benign client

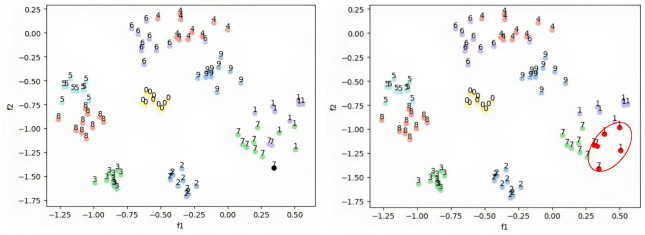


(b) Random poison

(c) Target poison, targeted class

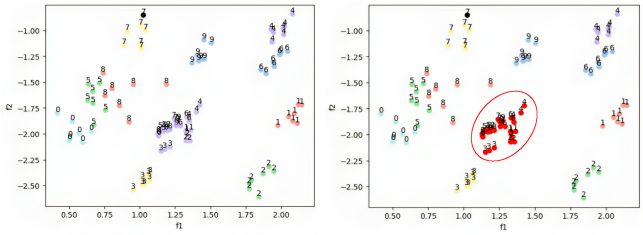
(d) Target poison, other class

Figure 7: SHAP feature attributions outputs for the 10 classes of MNIST. Benign clients have high feature importance and prediction probability in the class at position 3, which is not shown for the randomly poisoned client. In target poison, the poisoning client shows high prediction accuracy for target class 3 even though the label is 6, and SHAP feature attributions show scattered feature importances due to the impact of poisoning.



(a) 30% poisoners (b) Poisoners highlighted

Figure 8: Targeted poisoning scenario with 3 clients out of 10 are poisoned with 1 and 7 label swap poisoning. The highlighted elements show the poisoned clients have incorrectly classified the labels 1 and 7, yet their feature attribution clusters indicate the original values.



(a) 30% poisoners (b) Poisoners highlighted

Figure 9: Random poisoning scenario with 3 clients out of 10 are poisoned. The highlighted clusters show that poison feature attributions tend to form a different cluster than the normal distribution of clusters representing each class.

clustering, it can be observed that the clusters are correctly identified for each class by grouping the clients belonging to the same class into the same cluster, except for the poisoned classes 1 and 7.

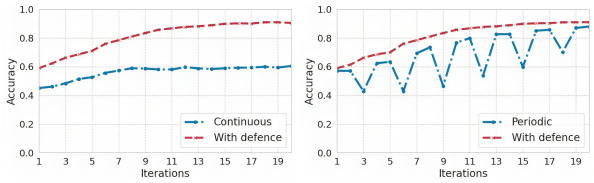
Unlike other clusters, which have the same class label for all points, we observe that the feature attributions of the same cluster have two labels for the two clusters corresponding to labels 1 and 7. Since we performed the label poisoning only for the two classes, the decision boundaries of other classes remained the same, which is evident by the correct feature attributions with similar probability densities as they were clustered correctly. However, for the 3 poisoned clients, the decision boundaries for the target classes have shifted such that 1 will be incorrectly classified as 7 and vice versa. The feature attributions show this since they belong in the wrong cluster, such that the feature attributions corresponding to output label 1 are considered the attributions related to label 7. This anomalous behaviour shown by feature attributions is the primary technique used in our algorithm to identify and eliminate poisoners.

When we consider the behaviour of clustering for a random poisoning scenario, we observe that all feature attributions of poisoners are detected as a separate cluster, as shown in Fig. 9. When randomly poisoning all the classes, SHAP feature attributions of the poisoners will be significantly different from benign clients, and since all the trained labels are random, all the feature attributions of poisoners will show similar probability densities. Thus, all feature attributions from the poisoners are detected in the same cluster. In this cluster, the labels are different, making it clearly anomalous than the other clusters, which can be detected by the detection algorithm. The poisoners are detected based on this behaviour of feature attributions, which can be provided as evidence. Therefore, the explainability of the detection here implies the ability to provide justifications by directly showing the suspicious feature attributions rather than relying on variations in the black-box AI models.

5.3. Evaluation of SHERPA

5.3.1. Elimination of Poisoning. We implemented the detection and defence mechanisms for multiple scenarios of poisoning. First, we consider the impact of poisoning on the accuracy of the FL global model. For the poisoning, we consider random poisoning type since it creates a higher impact on the model's decision boundary, thus degrading the overall model accuracy. Fig. 10 represents the impact of random poisoning on the CelebA dataset over 20 global FL rounds.

An adversary can perform poisoning continuously over FL rounds, or they can skip certain rounds and perform periodic poisoning, keeping intervals without poisoning between two consecutive poisoning rounds to make it more difficult for the detection algorithm to identify poisoning. However, when analysing the impact of poisoning, continuous poi-



(a) Continuous poisoning (b) Periodic poisoning

Figure 10: Comparison of poisoning impact on accuracy for continuous vs. periodic poisoning vs. poisoners removed.

soning has more impact on the aggregated model accuracy than periodic poisoning by skipping 2 rounds. In periodic poisoning, the decision boundary may get corrected back when poisoning is not done, but that is difficult if poisoning is continuously done. However, with our approach, both continuous and periodic poisoning are avoided as our robust algorithm can detect and eliminate the poisoners before the final aggregation.

We also implemented our defence for datasets MNIST, Fashion, NSL-KDD, and 5G-NIDD. Table 1 shows the results based on the FL system with 20 clients, where 10 are comprised of random poisoning and run for 20 rounds. The results show that the FL main task accuracy drops with poisoning when no defence is in place. However, with the defence, the elimination of poisoners with our algorithm has improved the accuracy close to the benign performance of the system.

TABLE 1: FL main task accuracy without poisoners, with poisoners and when poisoners are eliminated.

Dataset	FL acc.	No defence	With defence
MNIST	0.875	0.769	0.873
Fashion	0.714	0.666	0.708
NSL-KDD	0.957	0.938	0.953
5G-NIDD	0.994	0.926	0.989

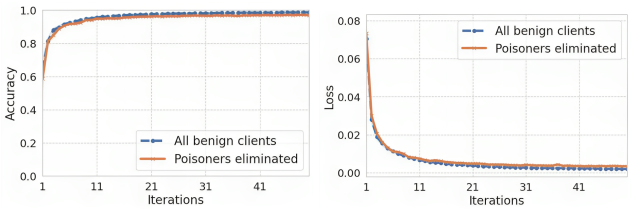
As the sample data d_s, we used 1 random input image for the MNIST and Fashion to obtain the SHAP values. In cases like random poisoning, a poisoned model deviates from benign models for any class. Thus, one input image can provide reasonable accuracy in detection. We later experimented with multiple sample data, which is discussed in Section 6.1.3. For NSL-KDD and 5G-NIDD, we used 10 sample data values for d_s since more values can provide better accuracy in case of binary classification scenario, also discussed in Section 6.1.3.

TABLE 2: Accuracy of the defence in detecting poisoners.

Dataset	FL Acc.	Def. Acc.	TPR	TNR	FPR	FNR
MNIST	0.873	0.950	0.900	1.000	0.000	0.100
Fashion	0.708	0.930	0.920	0.940	0.060	0.080
NSL-KDD	0.953	0.868	0.980	0.756	0.244	0.020
5G-NIDD	0.989	0.848	0.860	0.836	0.164	0.140

Table 2 presents the performance of the defence mechanism in correctly detecting poisoners from benign clients. Both scenarios have more than 90% defence accuracy at the 50% poisoning level.

5.3.2. Impact of Defence on the Accuracy. Eliminating suspicious clients will reduce the overall contributions from some clients. However, since they are highly likely to be poisoned, aggregating these contributions will eventually result in worse accuracy for the test predictions. We experimented with how the accuracy of the system behaves when the poisoners are removed, compared to a scenario where there are no poisoners in the system. Fig. 11 shows the average accuracy variation of 10 experiments of an FL system with 10 clients trained with the MNIST dataset for 50 FL aggregation rounds each, where 50% of clients are malicious and eliminated, compared to a system where all clients are benign.



(a) Accuracy (b) Loss

Figure 11: Main task test accuracy and loss of the aggregated model without poisoning vs. with poisoners eliminated.

The results show that when eliminating the clients, the resulting accuracy is also similar to the original test accuracy without poisoning even after removing 50% of the malicious clients. This resulting accuracy is similar to a scenario where 5 honest clients are aggregating the model. Thus, with more client participation, higher accuracy levels can be maintained even after eliminating a portion of suspects.

5.3.3. Comparison with other Robust Algorithms. We evaluated the detection accuracy of poisoners of our approach compared to existing poisoning techniques Multi-Krum [13], FoolsGold [14], and MOAT [17] for poisoning of MNIST dataset. We set 20 total clients where the poisoner percentages ranged from 20% to 80%. The results are provided in Table 3.

Our approach is applicable to various ranges of poisoning where the defence strategy can identify 98% of poisoners even when the majority of 80% of clients are poisoned. Our method also provides full coverage for low poisoning percentages where the algorithm can identify all poisoners compared with other approaches. The levels of false negatives are lower in our method, such that detection of poisoners as benign is minimal compared with others, which is more critical as the utility of the FL model can get compromised if poisoned models are aggregated.

5.4. Poisoning-based Privacy Attacks and Defences

Poisoning attacks are used as a key step in achieving better accurate results in privacy-related attacks such as property inference. We tested this by launching a type of PAPI attack [6] on the FL system. In the PAPI attack, the

TABLE 3: Poison detection accuracy of SHERPA compared to existing approaches.

Method	Metric	10% poisoned	30% poisoned	50% poisoned	80% poisoned
Krum [13]	TPR	0.800	0.533	0.550	0.750
	TNR	0.970	0.800	0.550	0.000
	FPR	0.030	0.200	0.450	1.000
	FNR	0.200	0.467	0.450	0.250
	Acc	0.960	0.720	0.550	0.600
FoolsGold [14]	TPR	0.000	0.200	0.300	0.763
	TNR	1.000	0.970	0.940	0.170
	FPR	0.000	0.030	0.060	0.030
	FNR	1.000	0.800	0.700	0.237
	Acc	0.900	0.740	0.620	0.780
MOAT [17]	TPR	0.400	0.267	0.440	0.700
	TNR	0.700	0.771	0.420	0.500
	FPR	0.300	0.229	0.580	0.500
	FNR	0.600	0.733	0.560	0.300
	Acc	0.670	0.620	0.430	0.660
Ours	TPR	1.000	1.000	1.000	1.000
	TNR	1.000	1.000	1.000	0.900
	FPR	0.000	0.000	0.000	0.100
	FNR	0.000	0.000	0.000	0.000
	Acc	**1.000**	**1.000**	**1.000**	**0.980**

attacker aims to identify if a certain property in the training dataset occurs at a particular FL training round. For this, the attacker injects poisoning samples where only the data records with the target property are randomly poisoned, while other records are kept in their original form. To implement the attack, we selected the dataset CelebA, which consists of facial images of people, along with its given binary attributes like gender and hair colour. For the model's main task, we selected gender classification from images. However, we selected the black hair colour as the target irrelevant property to be identified by the attacker, such that if any model is trained with facial images with the property of black hair, it will be detected by the attacker when analysing the global model after each FL round. An attacker may also use property inference to reveal the identity of a person participating in the FL training process from the client side. This can be invoked by selecting multiple properties for the attack that are specific to a targeted individual.

The attacker consists of samples containing the target property black hair, and they poison the gender variable in those records by target poisoning the existing value to $gender = male$. The system consists of 10 total clients, and we used 3 poisoning clients, where each client contained 1,000 poisoned records. When performing the aggregation by including the poisoners, we observe a drop in accuracy in the iterations where the target property is appearing in other benign clients periodically, as shown in Fig. 12.

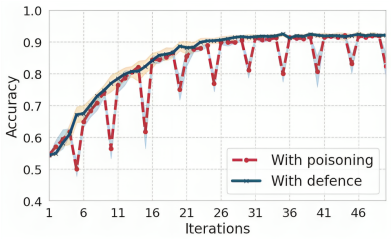


Figure 12: Accuracy of the property inference attack with poisoners vs. poisoners eliminated.

The inclusion of poisoners alters the decision boundary for the target class, which will be corrected by the honest clients. This alteration of decision boundary correction is reflected as the key information that is used for the property inference attack. We also implemented an attack model, which aims to automatically recognise this change for the attacker. This attack model inputs a set of flattened model parameters from a client and predicts if the target property is available in that client. For the training of the attack model, the attacker can use poisoners' local model updates from FL training, where the attacker knows that property is occurring in the poisoned models. The attacker can also include benign model updates without the property such that the dataset with and without the property can be made to train the attack model.

Since the inclusion of local models to the FL training by an attacker can be limited, we also use a set of shadow models where we create a set of 100 *shadow* models [6], [18] with the same NN architecture as the target models and each is trained for 10 rounds. The attack model is a random forest classifier with 50 estimators. After training the attack model, it is tested with the original aggregated FL models such that the attack with poisoning has a success rate of 85.2%. However, if the poisoners are eliminated from the FL system, it can be observed that the accuracy has dropped to 31.6%, which is lower than the random guessing accuracy. The results are shown in Table 4.

TABLE 4: Comparison of attack success with poisoning vs. after elimination of poisoners in the FL system.

Status	Accuracy	Precision	Recall	F1
With poisoning	0.852	0.860	0.840	0.850
No poisoning	0.316	0.160	0.500	0.240

Therefore, eliminating the poisoners from the FL system will make privacy attacks that are linked with poisoning more difficult, even without any other privacy-enhancing techniques. The aggregation process can itself act as privacy preservation, which does not cause any trade-offs like utility in the case of techniques like differential privacy [27].

6. Discussion

In this section, we consider various aspects of our robust algorithm and the factors that may affect the process. We also highlight the significance and novelty of our work compared with the state-of-the-art.

6.1. Factors affecting Clustering

6.1.1. Explainer Type. The clustering technique HDBSCAN can depend on the explanations that are input to the algorithm. The input data may depend on the explainer type, where there can be multiple methods to implement the SHAP feature attribution technique. We compared two techniques, deep explainer and gradient explainer, to evaluate how they may impact the overall clustering. Fig. 13 provides the accuracy of the overall convergence of the FL

system with 10 clients for MNIST, where 50% of the total clients are poisoners and run for 20 rounds.

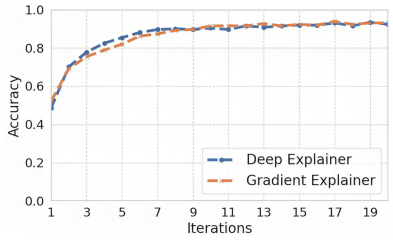


Figure 13: Comparison of the FL global model accuracy when using deep explainer vs. gradient explainer.

TABLE 5: Comparison of explainer type with the FL main task accuracy and defence accuracy.

Explainer	FL Acc.	Def. Acc.	TPR	TNR	FPR	FNR
Gradient Exp.	0.931	0.905	0.850	0.960	0.040	0.150
Deep Exp.	0.923	0.920	0.850	0.990	0.010	0.150

Here, we observe that both explainers have identified and eliminated poisoners in a similar manner, such that the FL global model accuracies of the two scenarios are 93.1% for gradient explainer and 92.3% deep explainer, shown in Table 5. Also, the defence accuracy is close to 90.5% and 92.0% for the gradient explainer and deep explainer, respectively. The process of SHAP is described in Equation 10 (Appendix A.2), where there are multiple implementations of techniques to calculate the SHAP values following the same equation. Therefore, the different implementations of SHAP can lead to similar results in detecting poisoners.

6.1.2. Configuration of Cluster Count. The output given by the clustering algorithm is then considered for poisoner detection, where the number of clusters may not necessarily be the same as the total number of classes unless parameters like cluster size or epsilon values are set. However, our proposed algorithm does not require the total number of clusters to be the number of classes. It only requires the information within a cluster, such that the clients belong to the same label class. Therefore, we can perform the detection conveniently with a minimal configuration overhead.

6.1.3. Detection Accuracy Improvements with Sample Reference Data. We evaluated the impact of the detection algorithm if the sample data size of d_s is modified. For this, we use MNIST dataset for multi-class classification and CelebA for binary classification scenarios. When considering the MNIST, we do not observe significant variation after $d_s = 2$, yet the computation time taken for calculations has significantly increased with higher d_s due to calculation of feature attributions each item in d_s. This is presented in Table 6.

The binary classification can result in fewer clusters since the number of classes is 2, and thus, a limited set of feature attributions will be available for clustering if the sample size is $d_s = 1$. To enhance the accuracy, we

TABLE 6: Comparison of Models with varying d_s for multiclass classification for MNIST.

Type	TPR	TNR	FPR	FNR	Accuracy	Time (s)
$d_s = 1$	1.000	0.846	0.154	0.000	**0.923**	0.525±0.074
$d_s = 2$	1.000	0.986	0.014	0.000	**0.993**	1.208±0.064
$d_s = 5$	1.000	0.996	0.004	0.000	**0.998**	2.568±0.099
$d_s = 10$	1.000	0.996	0.004	0.000	**0.998**	5.130±1.200

can increase the amount of sample data used for the reference set. When considering the application scenario gender classification with the CelebA dataset, we are required to implement binary classification. The classifier for local clients will output only two genders, male or female, as the main task output. For the robust aggregation, the SHAP feature attributions will, therefore, be the original output for 2 classes per each x_s sample data in d_s. With the increased amount of sample data at the reference set d_s, we observe the accuracy of the defence has also increased for a scenario with 10 clients, and 50% of them are poisoners. We ran the defence for 10 rounds and compared the accuracy of the poisoning client detection over the rounds. This is shown in Table 7.

TABLE 7: Comparison of Models with varying d_s for binary classification for CelebA.

Type	TPR	TNR	FPR	FNR	Accuracy
$d_s = 1$	0.600	0.543	0.457	0.400	**0.560**
$d_s = 5$	1.000	0.914	0.086	0.000	**0.940**

As shown from the results, for $d_s = 1$, we obtain a total accuracy of 56% for correctly detecting poisoners. However, with $d_s = 5$, this detection capability has increased to 94%. An increased number of sample data can increase the overall accuracy of the multi-class classification, but if the class count is high, the model will be accurate even at $d_s = 1$, as shown by the previous results. Therefore, this may be more useful with a small number of classes in the model.

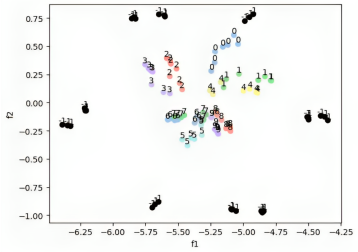


Figure 14: Differentiating poisoners for binary classification.

We also analyse the t-SNE plot for the gender classification of the distribution of SHAP feature attributions of client models. Fig. 14 shows that clients are scattered into multiple clusters, even though they have only two output classes. However, their feature attribution values corresponding to a class remain the same. In this scenario, feature attribution value clusters result from the corresponding example representative data since each example data x_s can have a unique set of feature importance values. However, the -1 values denote outliers, and they are clustered in the same cluster by the HDBSCAN algorithm. These outlier clusters

represent the poisoners, which are detected by our algorithm since the feature attributions are conflicting with each other.

6.2. Performance Costs

The computation time for SHAP feature attributions is considerably higher when compared with simple distance metric calculations like cosine similarity, which is often used in robust algorithms, since currently, the attributions for a given feature have to be calculated for all combinations of features to determine how important that feature is. However, the technique is performed on a server rather than a client device and thus can have higher computation capability and resources. Further experimental details on the computation time are provided in Appendix D.2. Moreover, in a practical scenario, all clients may not send the models at the same time, and they may send the models periodically, from which the aggregator has enough time to compute the feature attributions for a particular client. In this case, the time consumption for the defence will not have an impact on the overall aggregation time, as the time for sending an updated iteration may require more time than the time that can take to evaluate the client model.

6.3. Comparison with Related Works

Recent advances in the robust algorithms in the context of FL can be observed to use clustering and similarity score-based approaches often. Some approaches are performing direct elimination of clients [17]; meanwhile, other approaches may penalise the clients based on the trust scores or distance metrics [14], [15]. However, none of the techniques we presented use a SHAP and clustering-based mechanism to detect poisoners that can identify and differentiate suspicious clients while creating the possibility of providing which features in a specific client are poisoned and what classes the poisoners are targeting. Our work can provide an investigation of the malicious intentions of the poisoner, where the defence mechanism can identify if the poisoning is performed on a targeted property or in a random manner. Table 8 summarises our contributions compared with the existing robust algorithms.

TABLE 8: Summary contribution of our work.

Characteristics	Ref [17]	Ref [14]	Ref [13]	Ref [21]	Ref [16]	**Our work**
Use of XAI SHAP-based poisoning detection	H	L	L	L	L	**H**
Clustering of client updates	L	L	L	L	H	**H**
Assessment of poisoner intentions	M	L	L	L	M	**H**
Suspicious scores assessment for poisoners	L	M	M	L	L	**H**
Justifications for poisoner elimination	L	L	L	L	L	**H**
Evaluation of privacy attacks with poisoning	L	L	L	L	L	**H**
Enhancing the explainability of the defence mechanisms	M	L	L	L	L	**H**
High accuracy of poisoning detection at high poisoning levels	M	M	L	L	M	**H**

H High: The paper considers this area in high detail.
M Medium: The paper partially considers this area, no specific focus.
L Low: The paper has no/very low consideration in this area.

7. Conclusion and Future Work

This research investigated the possibility of utilising SHAP-based post-hoc feature attributions as a novel defence against poisoning attacks on FL. We presented our robust algorithm, SHERPA, that assesses SHAP feature attributions from FL clients via the aggregator and uses HDBSCAN clustering to detect and eliminate poisoners. Unlike the existing approaches, we aimed to provide better justifications for considering a particular client update as a suspect of poisoning by observing the behaviour of class-based feature attributions of clients. Our results show that poisoning can be determined via feature attributions with higher detection accuracy than the existing benchmarks, along with better explanations of the targeted features that can identify the adversary's malicious goals and intentions. With the information, an organisation can assess the significance of the model information that is targeted by attackers and take action to eliminate future threats arising from such vulnerabilities. For future research direction, we aim to enhance the explainable AI techniques for more lightweight implementation based on minority attributions and use a new architecture of hierarchical FL-based aggregation to distribute computational load efficiently.

Acknowledgments

This research is partly supported by the European Union in SPATIAL (Grant No: 101021808), the Horizon Europe JU SNS ROBUST-6G (Grant Agreement no. 101139068), and Science Foundation Ireland under CONNECT phase 2 (Grant no. 13/RC/2077_P2) projects.

References

[1] 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*. PMLR, 2017.

[2] S. Zhang and D. Zhu, "Towards Artificial Intelligence Enabled 6G: State of the Art, Challenges, and Opportunities," *Computer Networks*, vol. 183, p. 107556, 2020.

[3] Y. Siriwardhana, P. Porambage, M. Liyanage, and M. Ylianttila, "AI and 6G Security: Opportunities and Challenges," in *2021 Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit)*. IEEE, 2021, pp. 616–621.

[4] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov, "Exploiting Unintended Feature Leakage in Collaborative Learning," in *2019 IEEE symposium on security and privacy (SP)*. IEEE, 2019, pp. 691–706.

[5] H. Hu, Z. Salcic, L. Sun, G. Dobbie, and X. Zhang, "Source Inference Attacks in Federated Learning," in *2021 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2021, pp. 1102–1107.

[6] Z. Wang, Y. Huang, M. Song, L. Wu, F. Xue, and K. Ren, "Poisoning-assisted Property Inference Attack against Federated Learning," *IEEE Transactions on Dependable and Secure Computing*, 2022.

[7] J. Zhang, J. Zhang, J. Chen, and S. Yu, "GAN Enhanced Membership Inference: A Passive Local Attack in Federated Learning," in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*. IEEE, 2020, pp. 1–6.

[8] Y. Huang, S. Gupta, Z. Song, K. Li, and S. Arora, "Evaluating Gradient Inversion Attacks and Defenses in Federated Learning," *Advances in Neural Information Processing Systems*, vol. 34, pp. 7232–7241, 2021.

[9] L. Zhu, Z. Liu, and S. Han, "Deep Leakage from Gradients," *Advances in neural information processing systems*, vol. 32, 2019.

[10] V. Tolpegin, S. Truex, M. E. Gursoy, and L. Liu, "Data Poisoning Attacks against Federated Learning Systems," in *Computer Security–ESORICS 2020: 25th European Symposium on Research in Computer Security, ESORICS 2020, Guildford, UK, September 14–18, 2020, Proceedings, Part I 25*. Springer, 2020, pp. 480–501.

[11] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo, "Model Poisoning Attacks in Federated Learning," in *Proc. Workshop Secur. Mach. Learn.(SecML) 32nd Conf. Neural Inf. Process. Syst.(NeurIPS)*, 2018, pp. 1–23.

[12] X. Yang, W. Luo, L. Zhang, Z. Chen, and J. Wang, "Data Leakage Attack via Backdoor Misclassification Triggers of Deep Learning Models," in *2022 4th International Conference on Data Intelligence and Security (ICDIS)*. IEEE, 2022, pp. 61–66.

[13] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer, "Machine Learning with Adversaries: Byzantine Tolerant Gradient Descent," *Advances in neural information processing systems*, vol. 30, 2017.

[14] C. Fung, C. J. Yoon, and I. Beschastnikh, "Mitigating Sybils in Federated Learning Poisoning," *arXiv preprint arXiv:1808.04866*, 2018.

[15] X. Cao, M. Fang, J. Liu, and N. Z. Gong, "FLTrust: Byzantine-robust Federated Learning via Trust Bootstrapping," *arXiv preprint arXiv:2012.13995*, 2020.

[16] T. D. Nguyen, P. Rieger, R. De Viti, H. Chen, B. B. Brandenburg, H. Yalame, H. Möllering, H. Fereidooni, S. Marchal, M. Miettinen *et al.*, "{FLAME}: Taming Backdoors in Federated Learning," in *31st USENIX Security Symposium (USENIX Security 22)*, 2022, pp. 1415–1432.

[17] A. Manna, H. Kasyap, and S. Tripathy, "Moat: Model Agnostic Defense against Targeted Poisoning Attacks in Federated Learning," in *Information and Communications Security: 23rd International Conference, ICICS 2021, Chongqing, China, November 19-21, 2021, Proceedings, Part I 23*. Springer, 2021, pp. 38–55.

[18] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership Inference Attacks Against Machine Learning Models," in *2017 IEEE symposium on security and privacy (SP)*. IEEE, 2017, pp. 3–18.

[19] S. Truex, L. Liu, M. E. Gursoy, L. Yu, and W. Wei, "Demystifying Membership Inference Attacks in Machine Learning as a Service," *IEEE Transactions on Services Computing*, vol. 14, no. 6, pp. 2073–2089, 2021.

[20] T. Nguyen, P. Lai, K. Tran, N. Phan, and M. T. Thai, "Active Membership Inference Attack Under Local Differential Privacy in Federated Learning," *arXiv preprint arXiv:2302.12685*, 2023.

[21] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, "Byzantine-Robust Distributed Learning: Towards Optimal Statistical Rates," in *International Conference on Machine Learning*. PMLR, 2018, pp. 5650–5659.

[22] L. Deng, "The MNIST Database of Handwritten Digit Images for Machine Learning Research [Best of the Web]," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[23] P. Eigenschink, T. Reutterer, S. Vamosi, R. Vamosi, C. Sun, and K. Kalcher, "Deep Generative Models for Synthetic Sequential Data: A Survey," *IEEE Access*, 2023.

[24] D. J. Beutel, T. Topal, A. Mathur, X. Qiu, J. Fernandez-Marques, Y. Gao, L. Sani, K. H. Li, T. Parcollet, P. P. B. de Gusmão *et al.*, "Flower: A Friendly Federated Learning Research Framework," *arXiv preprint arXiv:2007.14390*, 2020.

[25] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: A Novel Image Dataset for Benchmarking Machine Learning Algorithms," *arXiv preprint arXiv:1708.07747*, 2017.

[26] S. Samarakoon, Y. Siriwardhana, P. Porambage, M. Liyanage, S.-Y. Chang, J. Kim, J. Kim, and M. Ylianttila, "5G-NIDD: A Comprehensive Network Intrusion Detection Dataset Generated over 5G Wireless Network," 2022. [Online]. Available: https://dx.doi.org/10.21227/xtep-hv36

[27] M. Li, Y. Tian, J. Zhang, D. Fan, and D. Zhao, "The Trade-off Between Privacy and Utility in Local Differential Privacy," in *2021 International Conference on Networking and Network Applications (NaNA)*, 2021, pp. 373–378.

[28] Z. Yang, M. Chen, K.-K. Wong, H. V. Poor, and S. Cui, "Federated Learning for 6G: Applications, Challenges, and Opportunities," *Engineering*, vol. 8, pp. 33–41, 2022.

[29] S. M. Lundberg and S.-I. Lee, "A Unified Approach to Interpreting Model Predictions," *Advances in neural information processing systems*, vol. 30, 2017.

[30] Z. C. Lipton, "The Mythos of Model Interpretability: In Machine Learning, the Concept of Interpretability Is Both Important and Slippery," *Queue*, vol. 16, no. 3, pp. 31–57, 2018.

[31] S. Jesus, C. Belém, V. Balayan, J. Bento, P. Saleiro, P. Bizarro, and J. Gama, "How Can I Choose an Explainer? An Application-Grounded Evaluation of Post-hoc Explanations," in *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*, 2021, pp. 805–815.

[32] R. J. Campello, D. Moulavi, and J. Sander, "Density-Based Clustering Based on Hierarchical Density Estimates," in *Pacific-Asia conference on knowledge discovery and data mining*. Springer, 2013, pp. 160–172.

Appendix A. Preliminaries

A.1. Federated Learning

FL is a distributed ML technique that is designed with the aim of privacy preservation at the user level. FL comprises many clients and an aggregator, where the clients receive an initial model parameter w_1 from the aggregator at the FL aggregation round 1. A client can then run local training rounds for the received model using their private data. Local models are then shared with an aggregator. The aggregator combines many local model updates together to create a global model. The aggregation process can be made with a technique such as FedAvg [1], which is represented as;

$$w_t \leftarrow \sum_{k=1}^{K} \frac{n_k}{n_t} w_t^k \tag{8}$$

where the global model parameter w_t at round t obtained by getting the sum of all parameters of K clients where a client k has n_k data samples out of n_t total samples.

Then these aggregated global models are forwarded back to the client, starting a new FL aggregation round with the client. Throughout the process, client data remains within

the client device, ensuring the privacy of data by eliminating the requirement of transmitting it to a third party. Applications like future B5G/6G network-based services can be expected to highly utilise FL-based approaches to collaboratively train ML models for both third-party applications and AI-driven 6G services [28], which also make FL a target for adversaries to launch attacks against the model integrity and the privacy of users via poisoning attacks.

A.2. Shapley Additive Explanations

Understanding the behaviour of complex models like deep neural networks is difficult due to its large number of parameters and their dependencies. Therefore, a simpler explanation model is used to create a more interpretable approximation of the original model. One approach for creating such explanations is to map a relation between simplified inputs and an approximate output from the model. Additive feature attribution is such a technique, which uses a linear function to create a simple explainer model in the form of [29]:

$$g(z') = \phi_0 + \sum_{i=1}^{M} \phi_i z'_i \tag{9}$$

where $z' \in 0, 1^M$ with M being the number of simplified input features, and ϕ_i is the attribution of feature i. The SHAP technique can be used to derive these feature attributions.

SHAP [29] is regarded as an XAI method which assesses the contributions of features of an ML model via its outputs. Since SHAP is used after the model is trained as a separate technique to *reverse-engineer* and understand model behaviour, it is considered a post-hoc XAI method [30]. It provides the resulting attributions for each feature by individually evaluating the importance of the presence or absence of these particular features. Feature attribution score ϕ_i for a target feature i is obtained by calculating its combinations of subsets of features with and without the target feature i. This is represented as:

$$\phi_i(f) = \sum_{S \subseteq F \setminus \{i\}} \frac{|S|!(|F| - |S| - 1)!}{|F|!} [f(x_S \cup \{i\}) - f(x_S)] \tag{10}$$

where F is the set of all features and $S \subseteq F$. The model f is first trained with values x_s with the features from S and including feature i. Then, it is trained by excluding i. The difference is obtained between the two, and this is repeated for all possible sets of S excluding i. The summation of all values results in the additive feature attribution for the target feature i. Work in [31] shows that SHAP can provide relatively high decision accuracy in a shorter time period for end users in real-world use cases, compared with other post-hoc XAI techniques like LIME or tree interpreter, apart from the original data, which is not accessible in FL. In this paper, we use these feature attribution values from SHAP as inputs to the algorithm, where we can get an understanding of the model behaviour and detect potential poisoners.

A.3. HDBSCAN Clustering

HDBSCAN [32] is a clustering algorithm that derives the probability distribution of the data to identify clusters within it. To approximate the probability distribution, the algorithm uses a metric called mutual reachability distance d_{mreach}, which is defined as [32]:

$$d_{mreach}(a, b) = max\{d_{core}(a), d_{core}(b), d(a, b)\} \tag{11}$$

where d_{core} are the core distances of two points a and b which are the distances of k nearest neighbours for each point. The other is the distance between the two points, which can be a metric like Euclidean distance. Based on the probability distribution with distances, the algorithm creates a hierarchy of clusters where high-probability points are placed in the same cluster.

Based on the clusters created, HDBSCAN generates a Minimum Spacing Tree (MST) at various probability densities indicated by distances among points in MST. Here, higher distances indicate low probability density between points. Fig. 15 shows an MST generated for the clusters generated for SHAP feature attributions of 10 models in an FL system where 4 clients are poisoners.

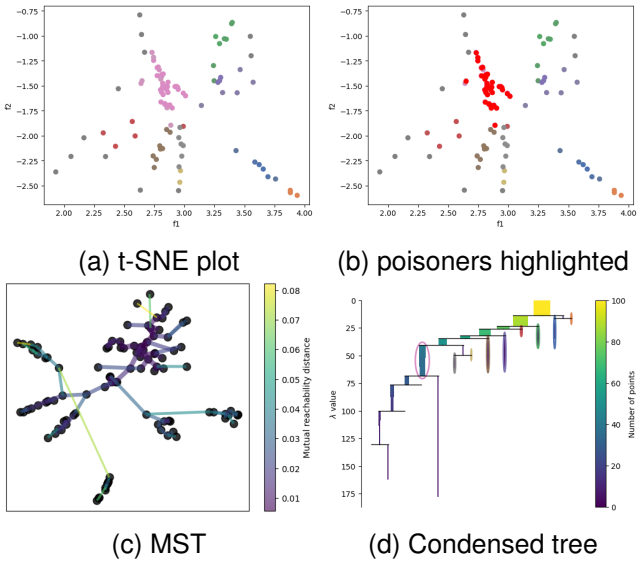


Figure 15: MST and condensed tree in HDBSCAN clustering graphs for an FL client analysis of a system with 10 clients with 4 randomly poisoned clients.

The algorithm then creates a condensed tree where the clusters are separated automatically based on the user-defined parameters, including $min\_cluster\_size$ and $min\_samples$.

Appendix B.
Comparison between model weights and predictions

B.1. Model weights

Suppose an NN model M with n layers is used to obtain a prediction Y. For the input layer, let the input for the model be X. The input layer propagates the input as:

$$Z^{(1)} = W^{(1)}X + b^{(1)}$$
$$A^{(1)} = \sigma(Z^{(1)}) \tag{12}$$

At a layer i, let the output from the previous layer $i-1$ is $Z^{(i-1)}$.

$$Z^{(i-1)} = W^{(i-1)}A^{(i-2)} + b(i-1)$$
$$A^{(i-1)} = \sigma(Z^{(i-1)}) \tag{13}$$

and the next layer i

$$Z^{(i)} = W^{(i)}A^{(i-1)} + b^{(i)} \tag{14}$$

Suppose the weight vector of a new model $\tilde{M}$ is different by a set of Δ values at layer $i-1$. Then, the output from layer $i-1$ is

$$\tilde{Z}^{(i-1)} = (W+\Delta)^{(i-1)}A^{(i-2)} + b^{(i-1)}$$
$$\tilde{A}^{(i-1)} = \sigma(\tilde{Z}^{(i-1)}) \tag{15}$$

For two models M and $\tilde{M}$ to have equal outputs A at layer i, the value $Z^{(i)}$ and $\tilde{Z}^{(i)}$ should be equal. In such a case, we can get

$$W^{(i)}A^{(i-1)} + b^{(i)} = \tilde{W}^{(i)}\tilde{A}^{(i-1)} + b^{(i)} \tag{16}$$

$$\therefore \tilde{W}^{(i)} = \frac{W^{(i)}A^{(i-1)}}{\tilde{A}^{(i-1)}}$$
$$= W^{(i)}\frac{\sigma(W^{(i-1)}A^{(i-2)} + b^{(i-1)})}{\sigma((W+\Delta)^{(i-1)}A^{(i-2)} + b^{(i-1)})} \tag{17}$$
$$\therefore \tilde{W}^{(i)} = \alpha W^{(i)} \tag{18}$$

where α is the ratio between the outputs from $i-1$ layers in the two models M and $\tilde{M}$.

The cosine similarity between the two models at layer i can therefore be obtained as

$$cos(\theta) = \frac{W^{(i)}.\tilde{W}^{(i)}}{\|W^{(i)}\|\|\tilde{W}^{(i)}\|} = \frac{\alpha W^{(i)}}{\|\alpha\|\|W^{(i)}\|^2} \tag{19}$$

Here, the α value can be set such that the cosine similarity of layer i can lead to a very low value between the models M and $\tilde{M}$. However, their final predictions will be similar since the next layer, i, compensates for the difference. Yet, the cosine similarity between these two layers will also be low as the model weights are changed. The perturbations Δ can be applied for all layers such that the overall model $\tilde{M}$ will be significantly different from the original model M. Thus, measuring the similarity by directly applying it to model parameters may not always result in a more accurate depiction of the poisoning due to the stochasticity of the model update parameters when compared with the outputs from the models.

B.2. Model poisoning scenario evaluation

From the perspective of an attacker, the model updates may be manipulated via model poisoning attacks, such that the model parameters can be very similar, yet the predictions given by the poisoned model can consist of backdoors. Here, we present such an example scenario. For a NN model M, the output layer L can be represented as

$$Z^{(L)} = W^{(L)}A^{(L-1)} + b^{(L)} \tag{20}$$
$$Y = \sigma(Z^{(L)}) \tag{21}$$

where Y is the output prediction vector from the model M. Suppose an attacker copies the model update M as $\tilde{M}$. The objective of the attacker is to modify the last layer of the model $\tilde{M}$ such that predictions would result in the attacker's targeted output $\tilde{Y}$. In this case, the attacker can adjust the final layer's weights by following the procedure below.

$$\tilde{Y} = \sigma(\tilde{Z}^{(L)})$$
$$\tilde{Z}^{(L)} = \sigma^{-1}(\tilde{Y}) \tag{22}$$

If $\tilde{W}^{(L)}$ is the updated weight of the poisoned model, given that the activation function σ is either linear or it is possible to get an approximated inverse function, we can obtain

$$\tilde{Z}^{(L)} = \sigma^{-1}(\tilde{Y}) = \tilde{W}^{(L)}A^{(L-1)} + b^{(L)} \tag{23}$$
$$\therefore \tilde{W}^{(L)} = \frac{\sigma^{-1}(\tilde{Y}) - b^{(L)}}{A^{(L-1)}} \tag{24}$$

from Equation 20, we can derive that,

$$A^{(L-1)} = \frac{Z^{(L)} - b^{(L)}}{W^{(L)}} = \frac{\sigma^{-1}(Y) - b^{(L)}}{W^{(L)}} \tag{25}$$

substituting in Equation 24,

$$\tilde{W}^{(L)} = \frac{\sigma^{-1}(\tilde{Y}) - b^{(L)}}{\sigma^{-1}(Y) - b^{(L)}}.W^{(L)} \tag{26}$$

Therefore, we can consider that $\tilde{W}^{(L)} = \beta W^{(L)}$, where only the layer L is modified by β in the poisoned model, such that the output will result in poisoned $\tilde{Y}$. However, since all the other $L-1$ layers in the NN are unchanged, this can result in a high cosine similarity in the poisoned model $\tilde{M}$, compared to the benign model M. Thus, we can consider that similarity-based detection directly applied to model parameters may not provide resilience against crafted model updates that result in high model similarity. However, the output of the model can determine the effect of poisoning in such situations since the benign update Y is very different from the malicious update $\tilde{Y}$.

Appendix C.
Distance comparison with cosine similarity

For two normalised vectors a and b, the Euclidean distance between the two can be derived as $\|\mathbf{a}-\mathbf{b}\|$.

This can also be represented by the law of cosines as:

$$\|\mathbf{a}-\mathbf{b}\| = \sqrt{\|\mathbf{a}\|^2 + \|\mathbf{b}\|^2 - 2\|\mathbf{a}\|\|\mathbf{b}\|\cos(\theta)} \quad (27)$$

In case where Euclidean distance $\|\mathbf{a}-\mathbf{b}\|_1 > \|\mathbf{a}-\mathbf{b}\|_2$, we get,

$$\|\mathbf{a}\|^2 + \|\mathbf{b}\|^2 - 2\|\mathbf{a}\|\|\mathbf{b}\|\cos(\theta_1) > \|\mathbf{a}\|^2 + \|\mathbf{b}\|^2 - 2\|\mathbf{a}\|\|\mathbf{b}\|\cos(\theta_2) \quad (28)$$

Since the vectors are normalised, we get $\|\mathbf{a}\| = \|\mathbf{b}\| = 1$.

Thus,

$$2 - 2cos(\theta_1) > 2 - 2cos(\theta_2) \quad (29)$$

$$cos(\theta_1) < cos(\theta_2) \quad (30)$$

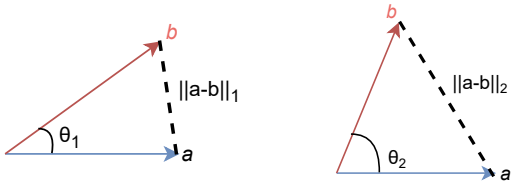


Figure 16: Cosine similarity of two vectors a and b with varying Euclidean distance

Therefore, the cosine similarity tends to decrease with the increase of the Euclidean distance between two normalised vectors. This shows that the higher the distances between two feature attribution vectors a and b, the lower the similarity between them and thus, they may fall into two separate clusters during the clustering process.

Appendix D.
Other Results

D.1. Accuracy with client count

We tested how our defence works with a varying number of total clients in the FL system, with 20, 50, and 100 clients, where 50% of the total clients in each case are poisoned. The defence accuracy has remained higher despite the increased number of clients, with significantly small false negatives of having undetected poisoners.

TABLE 9: Accuracy of the defence in detecting poisoners with client count for NSL-KDD.

Client No.	Def. Acc.	TPR	TNR	FPR	FNR
20	0.868	0.980	0.756	0.244	0.020
50	0.852	0.920	0.784	0.216	0.080
100	0.818	0.880	0.756	0.244	0.120

D.2. Time consumption of the defence

Table 10 presents the average time consumption in seconds to run a single FL aggregation round for different client configurations of 10 and 20 running with the MNIST dataset. The number of poisoners does not have an impact on time, leading to similar computation costs over varying levels of poisoning.

TABLE 10: Time consumption for MNIST under different poisoner counts.

Poisoner ct.	10 clients (s)	20 clients (s)
0%	8.903	15.990
50%	8.922	16.010
70%	8.893	15.980

Furthermore, we observed a linear increase in average time consumption for all the cases, where 20 clients consume approximately twice the overall computation time compared to 10 clients in the FL system. The runtime also varies with the dataset, as indicated in Table 11.

TABLE 11: Time consumption for the FL defence under MNIST and NSL-KDD datasets with 20 total clients.

Dataset	Def. Acc.	Total Time (s)	Avg. Time (s)
MNIST	0.923	16.010±1.873	0.800±0.419
NSL-KDD	0.868	1.308±0.130	0.065±0.029

Image datasets like MNIST may consume more time due to their higher number of features than NSL-KDD. Thus, the total time required for NSL-KDD per 20 clients and the average time computation required per single client are lower.

Appendix E.
Meta-Review

The following meta-review was prepared by the program committee for the 2024 IEEE Symposium on Security and Privacy (S&P) as part of the review process as detailed in the call for papers.

E.1. Summary

The paper presents SHERPA, a defense against poisoning attacks for Federated Learning models. SHERPA relies on SHAP values to identify if a client prioritizes relevant features for a given class. The feature attributions of client models are clustered (HDBSCAN) and then flagged if an infected model falls outside the expected cluster. The advantage of this methodology is that it not only flags poisoned client models but also explains why the client was marked as poisoned (e.g., not prioritizing features of the target class). Experiments show that even when 80% of the clients are poisoned, SHERPA still has an accuracy of 98%.

E.2. Scientific Contributions

- Creates a New Tool to Enable Future Science.
- Provides a Valuable Step Forward in an Established Field.

E.3. Reasons for Acceptance

1) The proposed methodology provides a valuable step in identifying poisoned clients for Federated Learning models.
2) SHERPA provides evidence as to why the model believes the client is poisoned, expanding on explainable AI (XAI).