Journal of Machine Learning Research 24 (2023) 1-50 Submitted 1/22; Revised 10/22; Published 1/23

Attacks against Federated Learning Defense Systems and their Mitigation

Cody Lewis CODY.LEWIS@NEWCASTLE.EDU.AU
Vijay Varadharajan VIJAY.VARADHARAJAN@NEWCASTLE.EDU.AU
Nasimul Noman NASIMUL.NOMAN@NEWCASTLE.EDU.AU
Advanced Cyber Security Engineering Research Centre (ACSRC)
The University of Newcastle, Newcastle, Australia

Editor: Sanmi Koyejo

Abstract

The susceptibility of federated learning (FL) to attacks from untrustworthy endpoints has led to the design of several defense systems. FL defense systems enhance the federated optimization algorithm using anomaly detection, scaling the updates from endpoints depending on their anomalous behavior. However, the defense systems themselves may be exploited by the endpoints with more sophisticated attacks. First, this paper proposes three categories of attacks and shows that they can effectively deceive some well-known FL defense systems. In the first two categories, referred to as on-off attacks, the adversary toggles between being honest and engaging in attacks. We analyse two such on-off attacks, label flipping and free riding, and show their impact against existing FL defense systems. As a third category, we propose attacks based on "good mouthing" and "bad mouthing", to boost or diminish influence of the victim endpoints on the global model. Secondly, we propose a new federated optimization algorithm, Viceroy, that can successfully mitigate all the proposed attacks. The proposed attacks and the mitigation strategy have been tested on a number of different experiments establishing their effectiveness in comparison with other contemporary methods. The proposed algorithm has also been made available as open source. Finally, in the appendices, we provide an induction proof for the on-off model poisoning attack, and the proof of convergence and adversarial tolerance for the new federated optimization algorithm.

Keywords: Federated learning, robust machine learning, on-off attack, Sybil, bad mouthing

1. Introduction

Federated learning (FL) distributes the training of the machine learning model over remote devices or endpoints, while keeping data localized. Training in heterogeneous and potentially large scale networks introduces novel challenges to machine learning, distributed optimization, and privacy-preserving data analysis. FL was first proposed as the federated averaging algorithm in Konečný et al. (2015), where each endpoint trains a model using its own data and periodically sending the model parameters (e.g. the gradients) to a server. The server aggregates these updates and sends back periodically the global model, which each endpoint then uses to update its own local model. This process helps the endpoints to obtain, with limited local data sets, a more generalized and robust model, without having

to share any of its local data to the server (and to other endpoints). This also helps to preserve the privacy of the local data at the endpoints, as they only share the updates to their models rather than their data. However an adversary still has the opportunity to learn about the data of the endpoints using only the model parameters (Shokri et al., 2017; Hitaj et al., 2017; Ateniese et al., 2015). Some approaches and techniques have been proposed to counteract against such attacks such as using differential privacy (Geyer et al., 2017) and security analysis of FL model attacks (Fung et al., 2018).

A large scale network involving many endpoints (and users) provides opportunities for adversaries to manipulate the model learning process, thereby adversely impacting the model of some or all endpoints. A notable example was the model in Microsoft's Twitter chatbot Tay, which was manipulated by malicious Twitter users in less than a day to learn and send offensive racist tweets (Mason, 2016; WakeField, 2016). Attacks on learning systems can also lead to more complex targets, such as recovering inputs that best fit a class in the model (Fredrikson et al., 2014, 2015) and stealing the model parameters or even the model itself (Tramèr et al., 2016; Reith et al., 2019; Joshi and Tammana, 2019). Different approaches and techniques have been proposed to address different types of attacks in FL. One aspect that is common to many of these approaches is the technique to detect anomalies.

In this paper, we investigate how these defense techniques in FL can be exploited through the use of network based attacks. We consider the use of on-off, bad mouthing and good mouthing attacks against FL models. The on-off attack exploits history aggregating anomaly detectors, whereas the good and bad mouthing attacks exploit the weighted averaging mechanisms in the global model update or even the subject of the update itself. The on-off attack is combined with other attacks, in which the on-off toggle allows the canonical attack to avoid mitigation. To achieve this, the adversaries toggle between attacking and acting honestly for periods of time. By doing so, the adversaries build up trustworthiness by acting honestly. Then while trusted, they perform their attacks until their trustworthiness diminishes below a threshold, when they toggle back to honest activity and repeat the process. On-off attacks have been demonstrated to be successful in a variety of applications such as in co-operative routing in MANET (Mao and McNair, 2010) and in directed diffusion based wireless sensor networks (Abuhaiba and Hubboub, 2013). The mouthing attack has two variants, good mouthing and bad mouthing, each having the goal of altering some victims' influence in the system. In good mouthing attacks, the adversary increases the influence of targeted victims, whereas in bad mouthing attacks their influence is decreased. Adversaries select their updates to the system in such a way that they make the victim look better or worse than it actually is.

1.1 Challenges and Motivations

Federated learning (FL) has been proposed to work with large scale distributed data environment on the one hand while maintaining the privacy of sensitive data on the other. It trains a generalized model, without the sharing of local data and therefore helps to preserve not only data privacy but also potentially helps to save network bandwidth. More importantly, as the storage and computational capabilities of remote devices in distributed networks have grown, it has become possible with FL to leverage local resources to train models directly

on remote devices. The FL setting brings out new challenges at the intersection of different fields such as challenges in privacy, large-scale machine learning, and distributed optimization. For instance, there is a need to protect the data generated on each device (model updates such as gradient information) when they are being communicated over the network either to a third-party, or to the central server. Approaches to protecting the privacy of federated learning can lead to reduced performance or system efficiency, and balancing these trade-offs can be a considerable challenge in realizing private federated learning systems. FL architecture introduces new threats and attack surfaces. One can envisage three types of attackers in FL systems, namely clients or remote devices, the aggregator and outsiders. An adversary can also assume a mixture of these roles and capabilities when attempting to have an adverse impact on the global model during training. For instance, as a client, the attacker can modify or replace the resulting local model's parameters before submitting it to the server, or can control and change the local training procedure such as the learning rate parameter, the local batch size, and the number of local epochs. As an aggregator, the attacker can alter the parameters of the global model, and examine model updates from clients and inject an attack in the aggregation algorithm. As an outsider, the attacker can intercept and modify communications between entities in the FL system. There can also be subtle attacks related to machine learning models. For instance, an adversary may not only aim to alter the resulting predictions from the machine learning model, but also make the model reveal additional information with its predictions. As the data being exchanged in the federated learning corresponds to the parameters of the individual models of the endpoints, the adversaries can attack by making appropriate alterations to their models in such a way that it can have the intended impact on the global model. In addition, classical network communication-based attacks, such as illegal alteration of data and replay, still remain in federated learning systems. In this paper, our focus is on the former attacks that exploit the characteristics of federated learning systems.

Several papers have proposed attacks and strategies to counteract attacks in FL systems. For instance, in backdoor attacks (Gu et al., 2017), the adversaries contribute gradients from compromised models and cause the global model to perform a chosen prediction when the sample it is evaluating contains a trigger (namely the backdoor). Alternatively, by contributing other *poisoned* gradients, the adversary may re-create a victim's data by tricking the other endpoints to train their models as the discriminator portion of a Generative Adversarial Network (GAN) (Hitaj et al., 2017). FL is also susceptible to free rider attacks (Lin et al., 2019), where the adversaries use the system without making their own contributions. For instance, they may calculate false gradients without any local training of their models.

Several approaches have been proposed to mitigate the attacks against the FL systems. For instance, Krum (Blanchard et al., 2017) and FABA (Xia et al., 2019) use the distances between each of the endpoints' updates to determine the faulty ones (based on the Byzantine problem). Others combat Sybil attacks (Fung et al., 2020) using the angles of the updates of each endpoint and finding potential collection of Sybils. Then there are approaches such as in Lin et al. (2019), which use additional machine learning mechanisms to identify which updates sent from endpoints are anomalies. Our paper proposes new attacks against FL systems that evade the above proposed mitigation approaches and mechanisms.

1.2 Our Contributions

Our paper has two main contributions: (1) We propose three types of attacks exploiting vulnerabilities in federated learning defense systems (FLDS) and present a detailed empirical analysis of these attacks under different conditions. (2) Then we propose a new federated learning algorithm that simultaneously mitigates all the proposed attacks, while at the same time remaining effective against previously proposed attacks in the literature. (3) We also provide an induction proof for the on-off model poisoning attack, and the proof of convergence and adversarial tolerance for the new federated optimization algorithm.

Specific contributions of our paper are as follows:

- **Three novel FLDS attacks**. *On-off label flipping attack* which allows an adversary to evade the existing mitigation strategies of FL defense systems. *On-off free riding attack* allows the adversary to free ride a system while evading detection mechanisms. *Good and bad mouthing attacks* allow the adversary to boost or reduce the targeted victim's influence on the system.

- **Demonstration of the proposed attacks.** We analyse and demonstrate these attacks on five federated learning algorithms, using three data sets of diverse characteristics, with two different machine learning models of different complexities under various test settings.

- **New mitigation strategy.** We propose a new federated learning algorithm that mitigates all our proposed attacks. We provide a proof of the convergence and effectiveness of this algorithm using optimization theory.

- **Implementation of mitigation strategy and simulations.** We have implemented the proposed new mitigation strategy and shown that they are successful at mitigating the attacks. The software implementation has been made available as open source.[1]

The rest of this paper is organized as follows. In Section 2 we review relevant related works describing the various attacks against machine learning models and their extensions to the federated learning environment. This section also discusses defense techniques proposed to counteract these attacks. In Section 3, we define the on-off label flipping attack, on-off free rider attack, and good and bad mouthing attacks against the FL defense systems. Section 4 describes the environment for our implementation, and outlines the choices for the design parameters and measurements. This section then discusses the experiments, results and analysis of the on-off label flipping attack, on-off free rider attack, and good and bad mouthing attacks. In Section 5, we propose our new federated learning algorithm and show that it is able to mitigate all the proposed attacks. Finally, Section 6 concludes the paper.

1.3 Notation

Throughout this paper, we consistently use the notations presented in Table 1.3.

1. `https://github.com/codymlewis/viceroy`

Notation	Description
p_u	Scaling factor of endpoint u's contribution within aggregation
$\mathcal{U}$	Set of endpoints
$\mathcal{H}$	Set of honest endpoints
$\mathcal{S}$	Set of Sybil adversary endpoints
$\bar{p}_Z$	Average scaling factor of the endpoints in the set Z
$\hat{p}$	Maximum attainable scaling factor w.r.t. the aggregation algorithm
θ_t	Global model at round t
$\Delta_{t,u}$	Update submitted at round t by endpoint u
$H_{t,u}$	History of updates submitted by endpoint u as of round t

Table 1: List of notations and symbols used in this paper

2. Related Works

The federated learning system provides new opportunities for the adversaries to exploit machine learning systems in different ways. Adversaries acting as malicious endpoints can manipulate their network-based interactions with the server to achieve their intended attacks.

2.1 Machine Learning and FL Attacks

In this subsection, we discuss the attacks against the machine learning (ML) models, by dividing them into four general archetypes similar to those mentioned in He et al. (2019). We will then analyse their potential impact on federated learning systems.

2.1.1 Poisoning Attacks

Poisoning attacks on machine learning involve inserting training data that causes the models to learn to perform the desired adversarial response. Biggio et al. (2012) presented this attack on support vector machines. Their results show how this attack can significantly impact the performance of the target model. These attacks can be used to achieve fairly complex goals, such as with the backdoor attack proposed in Gu et al. (2017). In this variant, malicious functionality is activated by some values in the data features, which is achieved via poisoned training data.

In a normal FL system, an adversary is not able to modify the data of other endpoints. That is, an adversary, as an endpoint, is only able to poison its own model. The act of poisoning its own model and then sending the corresponding gradients amount to the simplest form of the model parameter poisoning attack (Bhagoji et al., 2019). The authors additionally present how by scaling the gradients, the adversary can gain precedence for their updates over other endpoints in the system. This has been used in the backdoor attack in FL via parameter poisoning (Gu et al., 2017). The backdoor attack is a significant variant of model poisoning which aims to cause the global model to perform some adversary-chosen task only

when some trigger is present within the evaluated data sample, and to perform as normal when there is no trigger. This attack has recently been extended into two more sophisticated variations which to the best of our knowledge do not have effective methods for mitigation present in the literature. One of those two variants is the distributed backdoor attack proposed by Xie et al. (2019) which divides a backdoor trigger into small nearly undetectable partitions which are distributed across many adversaries. Where the combination of their attacks cause the trigger in it's complete form to effect the global model. The other variant is the edge-case backdoor proposed by Wang et al. (2020), which to great effect inserts the backdoor on samples that have minimal to no representation within the data of any of the honest endpoints within the system. Further demonstrating the impact of the edge-case backdoor, Wang et al. show its detection and therefore mitigation is an NP class problem, however, there is a caveat to this attack being that the backdoor is inserted on data that likely will not naturally appear to the endpoints.

Furthermore, Hitaj et al. (2017) present how the parameter poisoning attack can be used with GANs to generate a statistical copy of an endpoint's data. Here the adversary reconstructs the data of an endpoint by tricking the endpoint to train its model to discriminate images from the generator portion of the GAN.

Parameter poisoning attacks have also seen extensive study in the literature within a generalized form, Byzantine attacks (e.g. Blanchard et al. (2017); Xia et al. (2019); He et al. (2020)). These attacks see derivation from the Byzantine generals problem (Lamport et al., 1983), and consider a learning environment where out of the $3f + 1$ processed parameters up to f may be faulty. Faulty parameters are considered arbitrary and thus are inclusive of the variations of poisoning attacks, corruptions during communication, and/or various other corruptions that may occur within the system. Some works such as Xie et al. (2018) consider further generalizations of faults, where they may occur in an element-wise basis, enabling them to potentially pervade across all endpoints at arbitrary locations. In this work we consider only the parameter poisoning attack, referring to the Byzantine theorem in the classical sense where a robust system must be able to function with up to f faults, where faults are exclusively adversarial. We additionally consider more extreme situations with the Sybil attack which is considered as a more general network based attack and is discussed in Section 2.1.5.

2.1.2 Membership Inference Attacks

In the membership inference attack (MIA), an adversary determines whether a data sample is contained in the training data set, using the confidence of the model's predictions (Shokri et al., 2017). Their attack involves training a set of "shadow" models with the same hyperparameters as the target, on an arbitrary subset of data. The adversary may then form a data set of the model's predictions on the entire data set, which is then labeled based on whether the model has seen the data before. A meta model trained on the new data set can then be able to predict whether the target has seen some input before given the output it provides. The attack was later extended in Salem et al. (2019) by relaxing some constraints while providing equivalent results. The inference property proposed in Ateniese et al. (2015) can be thought of as a variation of MIA. It involves the adversaries determining whether the training data set has some statistical property of their choosing. They use a similar

structure involving a collection of shadow models that have been trained with or without the property, and a meta-learner is then trained on the set of shadow model parameters labeled with whether or not they were trained on a set with that property. The resulting model can then predict whether the target was trained on a data set with the property by taking the target's parameters as input.

In the federated learning context, this attack can provide the adversary with the average of the training data of endpoints in a specific class. Despite the observations presented in Jayaraman et al. (2020) stating that an environment similar to FL will make the attack ineffective, the attacks are still possible by exploiting the FL system's mechanisms. As adversaries have full access to the global model, they can perform ML-based attacks leading to inference on the average data across the endpoints. Statistical heterogeneity presented in Arivazhagan et al. (2019) may allow an adversary to make more detailed inferences about some endpoints using the global model, via biases created during training. This attack can be combined with the model stealing attack (Section 2.1.4) to target individual endpoints in the system.

2.1.3 Model Inversion Attacks

In these attacks, the adversary inverts the functionality of the model, and uses the predictions it provides to search for an input that best matches with a chosen class. The inversion operation involves the adversary choosing a target class, then searching for the input that makes the victim model return the greatest confidence in predicting the target class. Fredrikson et al. (2014) first proposed this attack using a brute force exhaustive search. Then they extended it in Fredrikson et al. (2015) using a gradient ascent based search.

Federated learning is designed specifically in such a way that the global model does not interact with the local data that is being used to train the endpoints. Similar to the MIA, this attack when performed on the global model will provide the adversary with an average of the data held by all of the endpoints in the system. Again, this attack can be combined with the model stealing attack (Section 2.1.4) to target individual endpoints in the system.

2.1.4 Model and Functionality Stealing

Another attack involves the adversary using the output from the target model and reproducing it by learning to act in the same way. This was first proposed by Tramèr et al. (2016) by equation solving. Here an adversary sends a set of inputs chosen in such a way that they form a collection of equations, which upon solving, yield the parameters of the target model. Machine learning is then used to make the attack more powerful; for example, Joshi and Tammana (2019) proposed an attack that applies this strategy to black box models using gradient descent so that it is dynamic to the perceived shape of the error function.

In the FL context, as the adversary has access to the global model, this attack can be used to recover the models of the individual endpoints. The effectiveness of this attack is dependent on the confidentiality of the communications between the endpoints and the server. This is because, in order to recover the model of a particular endpoint in the system, the adversary must intercept the gradients that specific endpoint sends. There are several techniques in the literature for preserving differential privacy, such as applying statistical

noise to updates (Shokri and Shmatikov, 2015; Wei et al., 2020; Li et al., 2020; Wang et al., 2019), homomorphic encryption (Hardy et al., 2017; Zhang et al., 2020), using secure protocols (Bonawitz et al., 2017) and other algorithms (Erlingsson et al., 2014).

2.1.5 Network-based Attacks

Certain network attacks can directly make use of the architectural characteristics and properties of FL. The Sybil attack proposed in Douceur (2002) involves the adversaries flooding the network with false identities to push the network state into their goal state. With federated learning, the Sybil attack may be used by an adversary to gain disproportionate influence over the global updates, without individually appearing anomalous. Conversely, adversaries could exploit the services provided by FL without contribution in the form of free riding. Lin et al. (2019) proposed an attack where instead of spending resources to contribute gradients to the global model, free riders may perform simple calculations to provide updates that have minimal to no impact on the resulting model, while maintaining a presence in the system.

Wan et al. (2021) proposed an attack that exploits additional information communicated in federated learning, such as the amount of data used in training in federated averaging, to achieve a more limited attack compared to the mouthing attacks that we have proposed in this paper. Their attack is able to give greater influence to the adversaries' updates over the global update than the updates from others, by claiming that they have been trained on great amounts of data, despite not doing so in reality. Conversely, they can give the other updates influence by communicating that the adversaries' updates have trained on a smaller amount of data than the other endpoints. The benefit of this attack is also its disadvantage, as it relies on the extra data communicated in the aggregation algorithm. On the one hand, it is often considered completely trustworthy, yet on the other hand, many algorithms do not use it. In contrast, the mouthing attacks that we propose in this paper remain in scope for all gradient/update aggregation algorithms by directly manipulating those updates rather than the additional data.

2.2 Federated Learning Defense Systems (FLDS)

To protect the FL systems from model poisoning attacks, changes have been made to the federated learning algorithms in the form of a modified update function. Generally, gradients of each endpoint, u, are scaled by an attack predictor value, p_u, rather than by the amount of training data of the endpoint. These systems may then locate the endpoints that submit poisoning gradients and reduce the scaling factor, thereby lowering the impact of these poisoning gradients. Such FLDS are often, but not exclusively, based on Byzantine fault tolerance proposed in Lamport et al. (1983).

Blanchard et al. (2017) proposed Krum and Multi-Krum algorithms that modify the FL to be resilient to Byzantine failures. Krum operates by selecting the received vector which is most similar to the $n - f - 2$ vectors closest to it. This corresponds to the Byzantine equation, where n is the number of endpoints and f is a parameter stating the number of faults expected to be tolerated. The multi-Krum variant selects the top m vectors that satisfy the Krum condition, where m is the parameter chosen by the user of the algorithm. Additionally, He et al. (2020) extended FL defense systems by presenting their susceptibility

to heterogeneous data, and proposed a re-sampling technique to adapt to the data. Both of these approaches can be interpreted as setting $p_u \leftarrow 1$ for the chosen endpoint (u) update vectors, and $p_u \leftarrow 0$ for the endpoint vectors that are not chosen.

As an alternative approach, some works modify the aggregation mechanism to exclusively include only the updates that sit inside the Byzantine condition. For example, Yin et al. (2018) proposed the co-ordinate-wise techniques of trimmed mean and median which respectively take the average of gradients with the largest and smallest $\frac{f}{2}$ elements removed, and take the median of the gradients. They observed that the trimmed mean approach achieves better performance, while the median approach eliminates the need for knowledge of the number of faults expected. This provides further motivation for finding techniques that average endpoint gradients without the need for knowledge of the number of faults, leading to learning approaches. Still requiring knowledge of the number of faults, Xie et al. (2018) extends the trimmed mean aggregation by taking co-ordinate-wise average of $n - f$ gradients nearest to the f-trimmed mean. They present that this extension enables better mitigation of the "generalized" Byzantine fault problem where faults can occur within subsets of each of the submitted gradients across all endpoints rather than being restricted the entire gradients of a subset of the endpoints.

The most commonly applied learning technique within FLDS is clustering by gradient similarity, primarily due to it's unsupervised properties and for the relaxation of the need for knowledge of the number of faults. For example, Muñoz-González et al. (2019) proposed a byzantine robust aggregation method which filters out updates based on their similarity to a scaled global update. If their similarity is further than a scaled standard deviation from the median similarity, then the update is removed. The updates are scaled into the global update based on a probability of being anomalous formed by a Markov model for each endpoint. Additionally, an earlier work, Auror (Shen et al., 2016), identifies key features among the data shared by endpoints by clustering each feature by endpoint and selecting those that have a distance between the clusters exceeding a parameterized value. Malicious endpoints are identified as those who have more than a parameterized percentage of key parameters not placed in the majority cluster.

Fung et al. (2020) proposed FoolsGold, a FLDS that goes further than Byzantine fault tolerance to resist Sybil attacks. They reduce the scale of updates based on the similarity of the angles of their gradients to each other. Hence they are still reliant on some consensus between the updates of the endpoints, not just on those that perform close to the same update. Awan et al. (2021) observed that many FLDS, including Krum and FoolsGold, make not only major assumptions about the adversarial populations, but also make assumptions about the distributions of data that endpoints hold. The robustness of such FLDS is reduced when these assumptions are not satisfied. So, Awan et al. proposed CONTRA which extends the FoolsGold algorithm with an additional scoring system that traces the activity of endpoints across the rounds.

Furthermore, other machine learning techniques may be applied to predict the gradients that are anomalous to the others. Lin et al. (2019) proposed the use of a deep auto-encoder Gaussian mixture model with an additional standard deviation measurement input (STD-DAGMM) to predict which of the endpoints are likely to be performing the free rider attacks. The model assigns each gradient an energy, and if the assigned energies are substantially higher than others, then they are likely to be free riders.

3. Threat Models and Proposed Attacks

In this section we discuss the environment, assumptions, and the threat objectives of the proposed attacks. We assume that the adversaries act as endpoints within the federated learning system and are capable of reading the gradient information submitted by other clients in the system. In more complex systems, this assumption can be relaxed with the use of cryptographic techniques such as those proposed in SecAgg (Bonawitz et al., 2017), or using Paillier encryption-based methods (Hardy et al., 2017; Zhang et al., 2020). However, the use of such mechanisms imposes certain limitations in the aggregation operations in the FLDS, thereby preventing the use of aggregation techniques that are robust to faults and parameter poisoning. In this paper, our focus is on the attacks that exploit those limitations in applicable gradient security to defeat the robustness of FL and proposing techniques for mitigating them. Our next work will address the transfer of encrypted gradients over the network and operations on encrypted data at the aggregation server. To enhance our proposed attacks, we assume that the adversary has knowledge of the federated learning algorithm being used and the chosen parameters in accordance with Kerckhoff's Law (Kerckhoffs, 1883a,b) and Shannon's maxim (Shannon, 1949).

In our experiments, we have a variable number of adversaries, each of which can perform the same attack, i.e. they work as Sybils (Fung et al., 2020). As our focus is on illustrating the dangers of adversaries' manipulating stateful or history-based systems, we have considered adversaries performing single task poisoning. Other works such as Nguyen et al. (2022) have considered multi-task poisoning against the FL defense systems, which are different from the attacks that we have studied. Our attacks specifically target the history building and prioritization systems being used, whereas multi-task poisoning targets flaws in the clustering system. Similarly, we do not consider stealthy model poisoning attacks (Bhagoji et al., 2019), which exploit the apparent data distribution for adversaries from the server's perspective. That is, they exploit i.i.d. data-based alterations to the loss function, which can make malicious gradients appear benign.

While we focus on the disadvantages of stateful-based robust FL, there are a range of benefits in their use, particularly within the per-silo environment which we consider in this work. Since per-silo federated learning is an environment where a relatively small number of endpoints with at least moderately sized data sets collaborate, it is expected that all endpoints contribute to each round and thus can be tracked by the server on an individual basis. The server can viably store information on each endpoint that may be used in aggregation to construct a more 'fair' system. That is, the server can store and use information built from gradients to better ensure that each endpoint has an equal contribution to the system. While the most obvious application within such a setup is to balance contributions towards a global goal (Karimireddy et al., 2020), it also increases the fairness of robust algorithms by reducing the impact of false positive identifications. By observing historical activity of each endpoint, the impact of a single fault or misidentification is less likely to completely condemn the endpoint in question. However, we will proceed to demonstrate the flaws of this approach and that a superior approach will need to strike a balance between reliance on historical and current-standalone information.

It is worth noting that the recording of states that we look at in this work does not alter the privacy preserving properties of the studied systems. We assume an environment where the

source of submitted gradients are identifiable (through e.g. IP or MAC address, etc.) as in most per-silo environments and as is possible within many cross-device environments. States are then only calculated using the received gradient information. As mentioned above, the gradients are not made private from the server (and unintentionally, the interceptor) to ensure the states can be calculated, as they are intended to provide a more fair yet robust system. However, in the rest of this paper, we will present how these states, when not carefully calculated, can be manipulated to the benefit of the adversary.

3.1 On-Off Label Flipping

The on-off attack exploits the system's building up expectation of the endpoints, based on how they have acted in the past. This is analogous to reputation in trust management systems. In the *On-Off Label Flipping*, the adversary acts as an honest endpoint for a period of time thereby causing the FL defense system to build up a positive expectation of the endpoint. This helps to strengthen the impact of the adversary's updates. The adversary then betrays the system and performs an attack for a subsequent period of time. This betrayal lowers the adversary's reputation. Then the adversary toggles back to being honest again after some time to recover its reputation. This On-Off attack loop is used to manipulate the system being exploited within the attack time windows. When there are multiple adversaries, they collude to determine when to toggle, so that they remain synchronised, that is, either all adversaries are on or all are off. When toggled on, any model poisoning or replacement attack is applicable. In our experiments, the adversaries apply the targeted model poisoning attack (Bhagoji et al., 2019), i.e. label flipping, while toggled on.

3.1.1 Adaptive Toggling of the Attack

In the on-off label flipping attack scheme, an obvious question to consider is when to toggle the attack? Since there are a number of elements (e.g. learning model, hyper-parameters, data sets etc.) that influence the best time to toggle, we have implemented a scheme that toggles the attack adaptively to maximize its effectiveness and evade detection. According to our assumption, the adversary intercepts the gradients from the other endpoints in the system, and then calculates $p_s : \forall s \in \mathcal{S}$, where $\mathcal{S} \subseteq \mathcal{U}$ is the set of Sybils that the adversary controls and $\mathcal{U}$ is the set of all endpoints. They determine whether the attack should be toggled on or off based on the average of the Sybil's influence, $\bar{p}_{\mathcal{S}}$, on the system as follows,

$$on_{t+1} = \begin{cases} false & : on_t \wedge (\bar{p}_{\mathcal{S}} < \beta \times \hat{p}) \\ true & : \neg on_t \wedge (\bar{p}_{\mathcal{S}} > \gamma \times \hat{p}) \\ on_t & : otherwise \end{cases} . \tag{1}$$

Equation (1) is a Boolean that determines whether the adversary should toggle "on" or "off" in the next time unit $t+1$. The resulting value of on_{t+1} represents the "on" status of the attack at the time $t+1$, where $true$ represents toggling "on" while $false$ represents toggling "off". This equation is a recurrence relation based on the toggle status during the current time unit, on_t. We additionally apply the unary operator $\neg$ to negate (bit flip) the value of on_t. Then there are the variables $\hat{p}$, β, γ and p used in the conditions of Equation (1). The

variable $\hat{p}$ represents the maximum possible influence the FL algorithm can assign (e.g. $\frac{1}{|\mathcal{U}|}$ for federated averaging, or 1 for Krum); the parameters β and γ are assigned according to the speed of change in the function p.
Using a grid search on a FoolsGold system with a network comprised of 50% adversaries, we found that $\beta = 1.0$ and $\gamma = 0.85$ are most effective. We, therefore, use those values in each of our experiments with continuous p values. In environments with discrete p, the change in the function is no longer observable, hence for those experiments we set a γ value that is greater than the Byzantine condition, $\gamma = 0.4$. Given that we were able attain significant results using these same parameters across all three evaluated data sets, and for differing number of endpoints within the system, it appears that these parameters are not dramatically sensitive to differing systems outside of the algorithm itself.

The benefit of using adaptive toggle is presented through our experiments in Section 4, where it is shown that the timing of toggles is dependent on the model hyper parameters, the federated optimization algorithm being used, the chosen parameter poisoning attack, and the data that has been used for training the model. We confirm this observation from,

$$\theta_{t+\tau} = \theta_t \prod_{i=1}^{\tau} \left(1 - \sum_{u \in \mathcal{U}} p_{t+i,u}\right) + \sum_{i=1}^{\tau} \sum_{u \in \mathcal{U}} \Delta_{t+i,u} n_{t+i,u} p_{t+i,u} \prod_{j=1}^{\tau-i} (1 - n_{t+j+1,u} p_{t+j+1,u}), \quad (2)$$

which shows the calculation of the global model at τ rounds ($\theta_{t+\tau}$) from that in tth round (θ_t). It describes how FL changes the model, θ_t, according to the starting point and the endpoint gradients and $\Delta_{t+i,u} : \forall i \in [0, \tau]$, with respect to time. We denote the learning rate of the endpoint's personal training of a model at time t as $n_{t,u}$, and τ as the number of units of time in the future for which the global model is evaluated. Despite the aforementioned dependencies for the effectiveness of the attack, we can also see from Equation (2) that the main impacting factor for the change in model over time is the gradient weighting mechanism. We can observe this by noting that all terms in the equation are multiplied by results from the weightings and that the other scaling factor, the learning rate, will be in most cases smaller than the expected value of the weightings. This supports our observation from our experiments in Section 4 and Appendix C that the parameters of our proposed attack's parameters have greater sensitivity to the weighting mechanism and a smaller sensitivity to other factors. The proof of Equation (2) in federated averaging systems is given in Appendix A.

3.2 On-Off Free-riding

We also propose an on-off attack with the free-riders, where adversaries train honestly for certain periods of time and then become free-riders in a subsequent period, and then continue with this toggling behaviour. By training honestly in a periodical manner, the adversary develops a history of being trustworthy for a period of time, which helps to avoid detection by history based defense systems during malicious periods. This allows the adversaries to obtain the benefits of the system while reducing their computational load compared to the other endpoints.
The literature presents three forms of free-riding, random weights, delta weights and advanced delta weights. These forms specify the adversary's strategy for calculating the

gradient, while in the free-riding state. These three forms of free-riding are described as follows (Lin et al., 2019):

- **Random weights**: Adversaries randomly generate their submitted gradient, $\Delta \leftarrow U(-10^{-3}, 10^{-3})$.
- **Delta weights**: Adversaries calculate the gradient as the difference between the current received global model and the previous one, that is $\Delta \leftarrow \theta_t - \theta_{t-1}$.
- **Advanced delta weights**: In addition to the delta weights calculation, the adversaries add noise to the submitted gradient, that is $\Delta \leftarrow \theta_t - \theta_{t-1} + \mathcal{N}(0, 10^{-3})$.

We apply on-off toggling to the delta weights variant to substantially enhance it's effectiveness. This selection is to best display the power of our proposed attack as delta weights balances ease-of-detection and number of computations. While the random weights variant is more trivial to detect due to being unlikely to resemble a true gradient, and advanced delta weights is the hardest to detect among the canonical free-rider attacks. However, we demonstrate that the on-off variant remains more effective.

3.3 Bad and Good Mouthing

Federated optimization algorithms use prioritization in defence systems to mitigate attacks. For example, the federated averaging algorithm prioritizes updates from an endpoint with $p_u = \frac{n_u}{N}$, where n_u is the number of samples the endpoint used to train this round, and N is the sum of the number of samples that all endpoints trained on. In our proposed bad and good mouthing attacks on FL, adversaries decrease or increase the impact of a victim's updates on the global model. They do so by selecting gradients that increase or decrease the priority of victim's gradients in the aggregation.
Generically, this attack can be defined as follows: as the adversaries know the chosen federated learning function, we have $F : \Delta \mapsto p$. Given the current set of gradient updates, Δ, the adversaries modify the gradients they send so that the resulting influence of the victim is either increased or decreased depending on the attack being performed.
To avoid detection, good mouthing and bad mouthing attacks can be made more resistant by adding noise ϵ to the gradients. In our experiments we have selected each element of ϵ in an i.i.d. manner from $\mathcal{N}(0, 10^{-4})$. This decision was made based on the work in Lin et al. (2019), which found that the same addition best aids in the masking of free-rider attacks. Therefore a mouthing attack can be performed as follows,

1. Select the victims gradient, $\Delta_{t,v}$.
2. Multiply the gradient by -1 if bad mouthing.
3. Add a noise vector to the gradient, $\Delta_{t,v} + \epsilon$.
4. Submit the gradient to the global model.

4. Experiments and Results

In this section, we first describe our approach for evaluating our proposed attacks, then we present the results of those described experiments.

4.1 System Description

We implemented a JAX (Bradbury et al., 2018) based simulation of a federated stochastic gradient descent system with 100 endpoints. All the endpoints and the server were simulated on a single machine. Each simulation involved training for 5000 rounds, where each round is a single epoch of training for each endpoint in the system, aggregating each of the new gradients in the server and sending the updated model to each endpoint.

4.1.1 Data sets

We performed our experiments on three data sets, namely MNIST (LeCun et al., 1998a), KDD cup '99 (Dua and Graff, 2017), and CIFAR-10 (Krizhevsky and Hinton, 2009). MNIST is a data set of 28×28 images of hand written digits. The training set contains $60,000$ samples and the testing set has $10,000$, each of which is class balanced. KDD cup '99 contains a set of network traffic flows labelled with 23 classes indicating attacks on the flows. The goal is to train an intrusion detection system. The training set has $345,815$ samples, and the testing set $148,206$, and the data is unbalanced with a strong bias towards samples in class "normal" and "Neptune" and "smurf". CIFAR-10 is an object recognition data set composed of 32×32 pixel images. It is a 10-class balanced data set with a training set of $50,000$ samples and a testing set of $10,000$ samples.

For experiments using the MNIST and the KDD cup '99 data sets, we trained a LeNet-300-100 network (LeCun et al., 1998b), while for the CIFAR-10 data set, we prepended a convolutional layer, with a kernel shape of 11×11 and a stride of 4, then a max pooling layer, with 2 strides, to the network.

We selected these three data sets to demonstrate distinct aspects in the evaluation of the proposed attacks. First, MNIST presents a simple task providing general intuition for our proposed attack's properties. Then, KDD Cup '99 presents a differing task in addition to imperfections in the data set, notably demonstrating the impacts of class imbalance. Finally, CIFAR-10 presents a more complex task and model, demonstrating whether the attack properties are retained in large scale federated learning objectives.

4.1.2 FL Algorithms

We have analysed our attacks against five federated learning algorithms namely FoolsGold (Fung et al., 2020), STD-DAGMM (Lin et al., 2019), Co-ordinate-wise Median (Yin et al., 2018)[2], CONTRA (Awan et al., 2021), and Multi-Krum (He et al., 2020). For FoolsGold, we set the confidence parameter $\kappa = 1$, as this variable indicates the average distribution of classes among the data per endpoint. For Multi-Krum, we have set the clipping value as the number of adversaries assigned during the experiment. This assignment provides the best possible performance of Multi-Krum, therefore making the hardest environment for attacks to succeed. For CONTRA, we set the number of expected honest endpoints to the exact number of honest endpoints and other parameters matching the original paper; this assignment also makes it the hardest environment for the attack to succeed.

2. For the rest of this paper, we refer to this algorithm as Median

In the case of STD-DAGMM, we have extended the algorithm to select multiple endpoints per update. First, "energies" are calculated by passing the endpoint's gradients through the STD-DAGMM model. Then, only the updates with energies in the range $[\mu-\sigma, \mu+\sigma]$ (where μ and σ are the mean and standard deviations of the current set of energies respectively) are selected. For the selected endpoints, we have performed a federated SGD update. This modified the STD-DAGMM to select approximately 68% of the received gradients.

4.1.3 Distribution of Data

To best demonstrate the effectiveness of the proposed attacks, we have selected the distributions of the MNIST and CIFAR-10 data sets, according to the observations provided in Awan et al. (2021). That is, for the multi-Krum, Median, and STD-DAGMM mechanisms, we distribute the data in an i.i.d. manner, and for FoolsGold and CONTRA[3], we distribute in a non-i.i.d. manner. This is due to the Byzantine and clustering mechanisms assuming that honest endpoints hold i.i.d. data since they identify adversaries based on apparent differences in the underlying data formed by the poisoning. FoolsGold, on the other hand, attains better performance in a non-i.i.d. environment, as its mechanism relies on punishing endpoints that submit gradients that are similar to others. Therefore, if the distribution of data across honest endpoints is too similar, then they will be falsely punished. For additional reference of the sub-optimal conditions, we present the other distributions, where relevant, in a 10 endpoint network in Appendix C.

For the MNIST and CIFAR-10 data sets, we use the Latent Dirichlet Allocation (LDA) (Hsu et al., 2019) method for data distribution. LDA assigns data to each endpoint such that no two endpoints hold the same sample, which ensures that our evaluations are not influenced by data replication. Furthermore, LDA is parameterized by a value, α, which uses the underlying Dirichlet function to determine the degree of data heterogeneity. That is, for larger values of α, the data is more i.i.d across endpoints, while for smaller values the data is more non-i.i.d. Therefore, we select the values of $\alpha = 0.5$ to model extreme heterogeneity when FoolsGold and CONTRA are evaluated; otherwise, we have set $\alpha = 1000$ to model extreme homogeneity when multi-Krum, Median and STD-DAGMM are evaluated.
For the KDD Cup '99 data set, we first remove the classes "spy" and "warezclient" as they are not present in the test data set. Then we assign each of the endpoints to one class of attack traffic along with the normal traffic resulting in five independent endpoints each holding the same 2 classes of data. For this data set, we evaluated the impact of class imbalance upon the success of the attack. We additionally assigned the normal traffic to each endpoint to make the environment more realistic and to prevent false penalization of honest nodes. This task demonstrates the middle ground in i.i.d. and non-i.i.d. distributions. However, our results in Sections 4.2-4.4 show that the model performance for this task is impacted significantly by the class imbalance issue, overruling any issues caused by data distribution at the endpoints. That is, the impact on the model performance is greater due to some endpoints holding a greater number of samples of their uniquely held class than other endpoints. This imbalance causes the level of non-i.i.d. aspect of the data to be a comparatively minor issue.

3. These assignments are not exclusively necessary for the Median and CONTRA algorithms, and hence we have selected them based on their functional similarity to the other algorithms.

4.1.4 Measurements

For our on-off label flipping and good/bad mouthing attack experiments, we have evaluated the attack success rate (ASR) of the global model on the test data set. ASR is the percentage of cases in which the specific attack achieves the adversaries goals. A higher ASR indicates that the attack is more effective, while a lower value indicates how effectively the attack is mitigated by the target. We tabulate our results showing the mean ASR and the standard deviation (STD) of the ASR, showing the amount by which the attack persists and it's variance according to the toggling. In these tables, a column labeled with "x% Adv." indicates that "x%" percentage of endpoints in the system are adversaries in that experiment.

Our specific methods of measuring the impact of each class of proposed attacks are given in the following paragraphs.

On-off label Flipping. We calculated the ASR based on the amount of targeted data that is labelled as intended by the attack. For example, if adversaries were attacking a model trained on MNIST, with the goal of making the global model labelling the digit 0 as 1, then the ASR would be the number of true 0s labelled as 1 in proportion to the total number of true 0s. Accuracy is often reduced by the success of the attack, as the induced labelling is usually not correct. This metric demonstrates the accuracy of the global model in performing the adversaries' target task.

Free Riding. The ASR for this case is calculated according to the influence, i.e. the p value, of the free riders indicating the performance of the defence mechanisms in detecting them. While the attack is toggled off, there are no free riders, so we have set the ASR to 0, so that the amount of system run-time that adversaries are free-riding is reflected in the mean and standard deviation of the ASR. On the other hand, when toggled on, the precise calculation for this metric is respective to the FL aggregation mechanism's format for assigning p values per endpoint, that is, whether $\sum_{u\in\mathcal{U}} p_u = 1$ or $\sum_{u\in\mathcal{U}} p_u = |\mathcal{U}|$. In the former case, the ASR calculated according to the mean of adversaries' p values divided by the fraction of endpoints with an p value greater than zero, therefore indicating the influence of the adversaries with respect to all endpoints that have some influence over the system. In the latter case, we calculate the ASR directly according to the mean of adversaries' p values as, for the respective algorithms, they are independent from each of other endpoints. This metric demonstrates as a ratio, how much the free-riders blend in the system relative to the other endpoints. Hence this value captures the system quality experienced by the free-riders despite the FLDS.

Good/Bad Mouthing. The ASR for bad mouthing is $\frac{d(\Delta_v,\Xi)}{\max_i d(\Delta_i,\Xi)}$ and for good mouthing is $\frac{\min_i d(\Delta_i,\Xi)}{d(\Delta_v,\Xi)}$, where Δ_v is the gradient of the victim, Δ_i is the gradient of the endpoint i, $\Xi = \theta_{t+1} - \theta_t$ is the aggregated global model update, and $d(a, b)$ is the Euclidean distance function. Without adversaries, the ASR would fluctuate around 0.5, as all updates with balanced data will make each gradient to be considered equally. This metric demonstrates the influence of the victim's update over the global update, therefore, capturing the amount by which their update is hindered in the bad mouthing case or boosted in the good mouthing case. We additionally indicate the cases where the victim and all adversaries have no influence over the system with $p = 0$ and with $p_{\mathcal{S}} = 0$ when only all the adversaries have

Environment	Algorithm	Mean ASR (STD ASR)		
		10% Adv.	30% Adv.	50% Adv.
MNIST	FoolsGold	0.241 (0.314)	0.725 (0.445)	0.993 (0.0828)
	Krum	7.7e-4 (4.8e-4)	4.87e-4 (0.0158)	5.9e-5 (3.75e-3)
	STD-DAGMM	0.0253 (0.0956)	0.889 (0.126)	0.988 (0.0169)
	CONTRA	0.199 (0.31)	0.458 (0.441)	0.539 (0.481)
	Median	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)
CIFAR-10	FoolsGold	0.989 (0.101)	0.995 (0.0694)	0.994 (0.0746)
	Krum	0.0684 (0.0682)	0.0594 (0.0541)	0.993 (0.0824)
	STD-DAGMM	0.0993 (0.0887)	0.855 (0.155)	0.936 (0.0836)
	CONTRA	0.261 (0.301)	0.57 (0.416)	0.961 (0.185)
	Median	0.0398 (0.0147)	0.0461 (0.0135)	0.0486 (0.0119)
KDD Cup '99	FoolsGold	0.999 (7.1e-4)	0.999 (0.0245)	1.0 (0.0193)
	Krum	1.0 (0.0)	1.0 (0.0)	1.0 (0.0)
	STD-DAGMM	0.999 (0.0245)	1.0 (0.0142)	1.0 (0.0137)
	CONTRA	1.0 (0.00359)	1.0 (0.0036)	1.0 (0.02)
	Median	0.999 (0.031)	0.999 (0.0285)	0.999 (0.0239)
Scaled Backdoor	FoolsGold	0.827 (0.378)	0.989 (0.106)	1.0 (0.0)
	Krum	1.3e-3 (0.0342)	1.2e-5 (1.25e-4)	2e-6 (9.1e-5)
	STD-DAGMM	0.987 (0.0449)	1.0 (6.8e-5)	1.0 (0.0)
	CONTRA	1.0 (9.25e-4)	1.0 (0.0127)	0.999 (0.0237)
	Median	0.00368 (0.0124)	0.126 (0.222)	1.0 (0.0125)

Table 2: Results of the On-Off Label Flipping Experiments

no influence over the system. We also indicate the cases where the adversaries induce an overflow in the global model parameters with $p \gg 1$.

4.2 On-Off Label Flipper Attacks on FLDS

We found that, especially for our FoolsGold and CONTRA experiments, the attack is most effective when adversaries spend their initial interactions honestly prior to toggling the attack on. Afterwards, the toggling acts to maintain the influence of the attack upon the system. This is due to their Sybil mitigating mechanism, which is based on the cosine similarities, and thus relative angles between the history vectors of each of the endpoints. These vectors are built by the aggregation of the endpoint's respective update gradients. Once the trusted history is built, the addition of adversarial updates has little impact on the relative angles between those resulting aggregated vectors, and hence the attacks are able to evade with minimal repercussions.

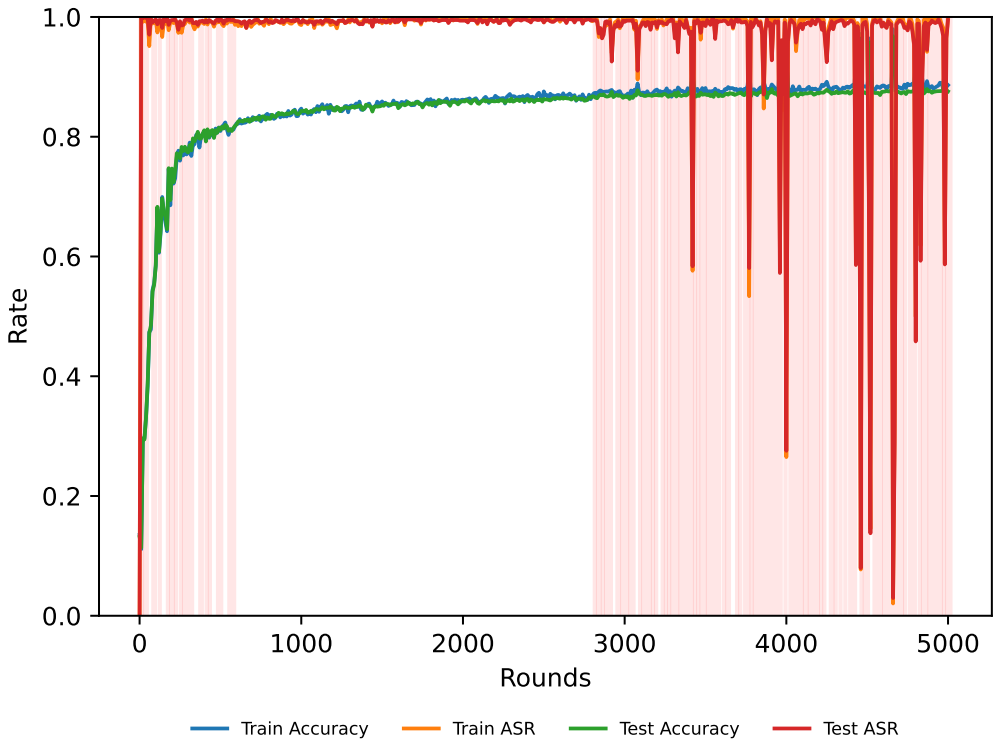

Figure 1: Measurements over the system run time of the 50% Adv. FoolsGold with 10 endpoints total. The red shading indicates periods of time where the attack is toggled "on". We observe that the on-off attack enables the sustenance of an high average attack success rate.

Here is a small example to illustrate this property. Let the update gradient be $\Delta = \langle -1, 1 \rangle$. With the history being empty, we have $H = \langle 0, 0 \rangle$. Then adding $H \leftarrow H + \Delta$ results in an angle of $\theta = \frac{3\pi}{4}$, whereas for a built trusted history, $H = \langle 50, 50 \rangle$, the resulting angle after adding the gradient becomes $\theta = 0.25637\pi$. In fact, it takes over 500 additions of that gradient for the resulting angle to converge to $\frac{3\pi}{4}$. After establishing trust, the endpoint is able to make large changes in its contributions without any repercussion. In the context of the example, an adversary after attaining the trusted state can send poisoning gradients that can still have a trusted influence until the history gets close to convergence to $\frac{3\pi}{4}$.
In Table 4.2, we present all the results from the experiments with the on-off label flipping attacks. In the MNIST environment, the FoolsGold, CONTRA and STD-DAGMM experiments demonstrate the clear trend between the amount of adversaries and the success of the attack. Figure 1 demonstrates an example of the attack's effects on the overall system run-time with a network of 50% adversaries in a FoolsGold system. The Multi-Krum and

Median experiments demonstrates a limitation to our proposed attack, which is its ineffectiveness against stateless defense mechanisms. However, in being stateless such mechanisms introduce their limitations, such as Krum's lack of Sybil defence, which is shown by the case where 50% of the endpoints are adversaries in the CIFAR-10 environment, or Median's inability to detect or mitigate scaled attacks as shown by the scaled backdoor environment. STD-DAGMM fails to mitigate this attack as it is designed only to detect whether a gradient is artificial. So the failure emerges from the poisoned gradients being produced by the adversaries still being generated through the training process as with honest endpoints. The difference being, when in the on state the adversaries train their model with poisoned data. There is a notable exception in the case where only 10% of endpoints are adversaries. The resulting ASR is not significant, as in this case STD-DAGMM is able to differentiate the gradients resulting from the adversaries' local models over the other honest ones. This is not the case when there are more adversaries, as at this point their updates are no longer the outliers that STD-DAGMM is searching for. Hence, STD-DAGMM only observes and accepts those legitimately generated gradients.

We demonstrate that the proposed attack remains effective within the more complex environment of learning the CIFAR-10 object detection using a convolutional neural network. The same general trends as in the MNIST experiments are observed, therefore showing that more complex tasks and larger models have minimal impact on on-off label flipping.

With our experiments with the KDD Cup '99 data set, we found that the balance of true class data points boosts the effectiveness of the attack. However, we observed and studied more unique properties of this environment for the 10 endpoint network, which we present in Appendix C.1.

We have compared the effectiveness of our attack with a state-of-the art scaled backdoor attack (Bagdasaryan et al., 2020). Our results show that on-off attack can achieve comparable performance for 30% or more adversaries. Our proposed attack remains beneficial by not requiring scaling, therefore being immune to simple norm clipping techniques due to having gradient magnitudes equivalent to the honest gradients. However, this means that by using norm clipping together with the median algorithm scaled, standard poisoning and on-off attacks may be mitigated. Despite these benefits, we find in Section 4.4 that the Median algorithm is particularly susceptible to the mouthing attacks. Hence our motivation still remains for proposing a unified mitigation algorithm in Section 5.

4.3 On-Off Free Rider Attacks on FLDS

The on-off free riding attack exhibits some of the same properties as the on-off label flipping attack. These shared properties are induced by the on-off mechanism to evade the history aggregation mechanisms of FoolsGold and STD-DAGMM. Ultimately, we show in Table 4.3, that the on-off variant of free rider attacks is mostly successful against FoolsGold, is mildly successful against Multi-Krum, is completely successful against STD-DAGMM, is moderately successful against CONTRA, and unsuccessful against Median. However, Multi-Krum and STD-DAGMM algorithms generally better mitigate the attack for a greater number of adversaries as the increase of adversaries skews the average of the updates thereby further exhibiting the free rider's uniformity to the average update. Conversely, the STD-DAGMM results are particularly surprising due to having been designed to mitigate free-riding at-

Data set	Algorithm	Mean ASR (STD ASR)		
		10% Adv.	30% Adv.	50% Adv.
MNIST	FoolsGold	0.982 (0.133)	0.243 (0.429)	0.0 (0.0)
	Krum	0.448 (0.454)	0.35 (0.355)	0.25 (0.253)
	STD-DAGMM	1.0 (0.0)	1.0 (0.0)	0.5 (0.5)
	CONTRA	0.308 (0.452)	0.383 (0.431)	0.403 (0.433)
	Median	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)
CIFAR-10	FoolsGold	0.976 (0.154)	0.962 (0.191)	0.802 (0.398)
	Krum	0.448 (0.454)	0.349 (0.354)	0.25 (0.253)
	STD-DAGMM	1.0 (0.0)	1.0 (0.0)	0.5 (0.5)
	CONTRA	0.287 (0.446)	0.388 (0.434)	0.41 (0.437)
	Median	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)
KDD Cup '99	FoolsGold	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)
	Krum	0.449 (0.455)	0.351 (0.356)	0.249 (0.252)
	STD-DAGMM	1.0 (0.0)	1.0 (0.0)	0.5 (0.5)
	CONTRA	0.131 (0.288)	0.0798 (0.177)	0.107 (0.202)
	Median	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)

Table 3: Results of the On-Off Free Riding Experiments

tacks. That is, as shown through mean ASR values above 0.5, the on-off toggling fools the STD-DAGMM learning mechanism, thereby rendering the defense mechanism ineffective.

FoolsGold is generally unable to completely mitigate the attack, due to the on-off attack exploiting the history mechanisms of the algorithm. Furthermore, we observe an interesting property in this on-off attack, from Figure 2. It shows that the effectiveness of the on-off free rider attack is dependent on whether the adversary starts in the "on" or the "off" state. If the adversary starts with the attack, that is, the "on" state, then the attack is completely mitigated, whereas if the adversary starts by being honest ("off" state), then the attack evades FoolsGold. This shows how FoolsGold inadvertently regards the first update from an endpoint to be the most important in determining the behavior. The history formed by adding the gradients from time 0 to the current time ensures that the first update has the greatest impact on the gradient vector's direction, whereas the successive gradient updates have progressively a minimal impact. If the adversaries collude and send the same update, then this leads to parallel update vectors being added to their history. Therefore, the angle of their history will steadily converge, and it is at this point the adversaries should toggle off.

We find an exception to the effectiveness of the attack against FoolsGold. This occurs with the unbalanced class representation in KDD Cup'99. In this case, the algorithm detects all the free riders despite on-off toggling. This emerges from the properties of the gradients

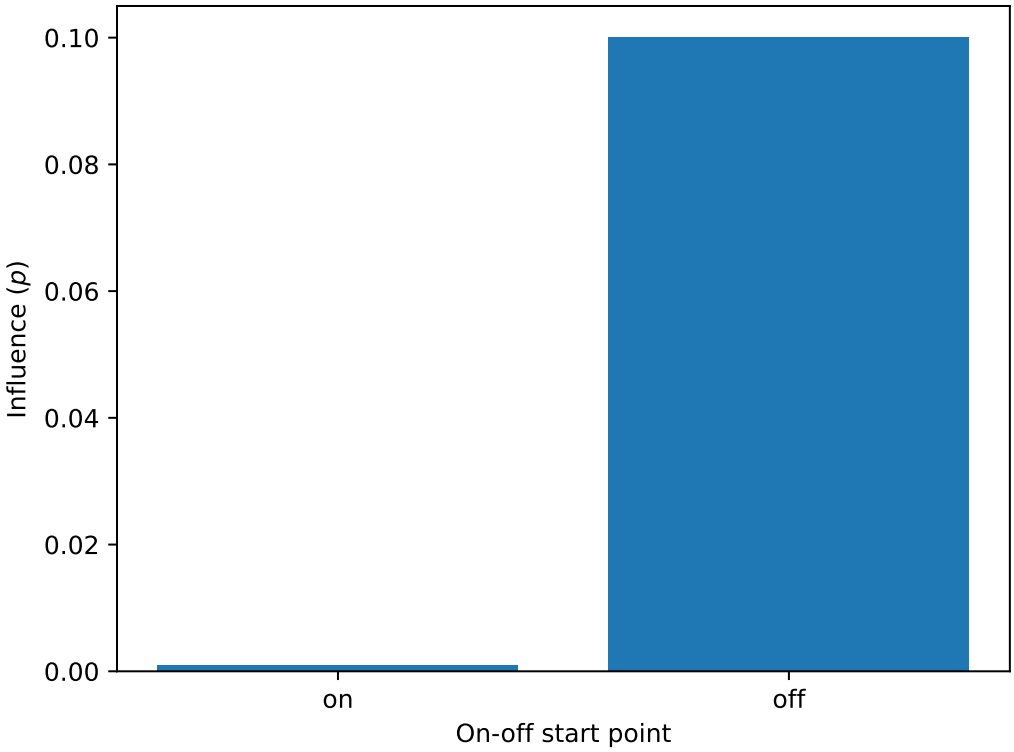

Figure 2: The FoolsGold performance is dependent on the toggle start states for two or more on-off free riders. To simplify this demonstration, we evaluated for a 10 endpoint network including 2 on-off free riders. We measure influence, which is the average measured p observed across the free riders at the final round of training.

generated in these cases. In this case, the adversaries' gradients are distinctly unique from any endpoint's and are therefore easily detected by the FoolsGold algorithm.

The attack generally has a moderate effectiveness against Multi-Krum, where the Byzantine collusion cases are more effectively mitigated. These patterns emerge through the functionality of Krum not being suited to the mitigation of free-rider attacks, as Krum merely selects gradients least distant from each of the other. Since the global update from the previous round will typically remain close to that of the current round's individual endpoint updates, the free-rider's will consistently evade the Krum algorithm. However, Krum remains relatively successful in detecting the collusion of Byzantine on-off free-riders; nonetheless it never achieves complete mitigation of the attack.

Despite being designed for the detection of free-rider attacks, we find the STD-DAGMM algorithm is most adversely effected by the on-off attack. Our results show that the impact

of the on-off toggling extends into systems that learn from historical gradients, that is, in addition to those that aggregate. A notable trend in the results emerges - the mean ASR generally being lower for 50% adversaries. This arises directly from our rendition of the algorithm only selecting 68% of the gradients at each round. Hence, the ASR recorded from the average of the adversary's influence at each round will be lowered simply due to some adversaries inevitably being selected due to having a large population.

The CONTRA algorithm is moderately impacted by the attack; yet unlike the other algorithms, in most cases the effectiveness of the attack increases with the greater percentage of adversaries. This occurs due to the scoring system used by CONTRA. When there are more on-off adversaries in the network, regardless of the attack type, the scoring system is fooled into disproportionately rewarding adversaries. By persistently switching between the on and off states, adversaries are able raise their score by not attacking, then toggle to an attack when the score is at or close to its maximum. More adversaries lead to more instances of the attack at any time and thus greater impact on the system. The exception is for the KDD Cup '99 environment, where due to use of mechanics derived from the FoolsGold algorithm, CONTRA more easily detects the distinct gradients generated in this environment, although there are some small impacts from the manipulation of the scoring system.

As we have seen in the previous section, variants of the on-off attack do not affect the Median algorithm; they are completely mitigated. This is again due to the median algorithm not being stateful. Additionally, it best mitigates free riders due to just selecting the median elements avoiding most of the skew that may occur from additional mean valued elements.

4.4 Good and Bad Mouthing Attacks

In Tables 4.4 and 4.4, we present the results from our experiments with good mouthing and bad mouthing adversaries. Our measurement of ASR works slightly differently compared to our previous experiments, as the expected value under no attack has small fluctuations close to 0.5. Another important factor where this attack differs from our other proposed attacks, is that this one does not use or require toggling. In lieu of that, the adversaries constantly perform their attack.

The FoolsGold algorithm effectively detects the adversaries and mitigates them. However, in the KDD Cup '99 environments this is to the detriment of the victim, as they are grouped together as Sybils with the adversaries. This flaw can be remedied by the application of a higher-level policy, at the cost of efficiency. For example, by eliminating the influence of all but one of the detected Sybils, the remaining endpoint can be determined to be a mouthing attacker or not, based on whether its next updates are a copy of some other endpoint's updates or are sufficiently unique (accounting for potential noise) within the system. Regardless, a major factor that enables the detection of the adversaries is the noise that is being applied to their updates. If present, this makes the adversaries' updates sufficiently different from victim but not enough from each other, since the noise is retrieved from the same distribution. FoolsGold performs similar effective mitigation for the bad mouthing attack.

The Multi-Krum algorithm was adversely effected by these attacks to a greater extent with respect to the number of adversaries. As Krum prioritizes gradients that are close to each

Environment	Algorithm	Mean ASR (STD ASR)		
		10% Adv.	30% Adv.	50% Adv.
MNIST	FoolsGold	$p_{\mathcal{S}} = 0$	$p_{\mathcal{S}} = 0$	$p_{\mathcal{S}} = 0$
	Krum	$p \gg 1$	$p \gg 1$	$p \gg 1$
	STD-DAGMM	$p \gg 1$	$p \gg 1$	$p \gg 1$
	CONTRA	0.15 (0.821)	0.216 (0.8)	$p = 0$
	Median	0.968 (0.027)	0.983 (0.0245)	0.986 (0.0222)
CIFAR-10	FoolsGold	$p_{\mathcal{S}} = 0$	$p_{\mathcal{S}} = 0$	$p_{\mathcal{S}} = 0$
	Krum	0.975 (0.211)	0.975 (0.112)	$p \gg 1$
	STD-DAGMM	$p \gg 1$	$p \gg 1$	$p \gg 1$
	CONTRA	0.182 (0.832)	0.237 (0.578)	$p = 0$
	Median	0.975 (0.0218)	0.994 (0.00867)	0.996 (0.00617)
KDD Cup '99	FoolsGold	$p = 0$	$p = 0$	$p = 0$
	Krum	0.631 (0.532)	0.752 (0.341)	$p \gg 1$
	STD-DAGMM	$p \gg 1$	$p \gg 1$	$p \gg 1$
	CONTRA	$p = 0$	0.217 (0.66)	$p = 0$
	Median	0.649 (0.388)	0.622 (0.145)	0.965 (0.0556)

Table 4: Results of the Good Mouthing Experiments.

other, the adversaries are given greater influence over the system, as the attack's basis is on copying the victim's gradient. Our results for the attacks on Krum show a complete success which arises from a flaw in the algorithm. As despite best efforts to prevent overflow within the Krum algorithm[4] itself, the mouthing attacks cause overflows in the parameters of the model due to complete failure to filter or reduce the scaling of the added updates. In the case of the bad mouthing attack, an adversarial search is created where as the adversaries add their negating gradients, and the victim responds through their learning process by producing larger gradients to better reset the model state. However, since this attack uses the victim's gradients, the adversaries' responses cause the victim to respond with even larger gradients thus creating a positive feedback loop from the adversarial perspective. The good mouthing cases, on the other hand, simply create progressively larger gradients due to the duplicates present causing progressive overshooting of the victim's target.

As with the case of the label flipping attack, STD-DAGMM is unable to detect or effectively mitigate the mouthing attacks. That is, due to the attack directly using the victim's gradient, which is legitimate. Therefore STD-DAGMM allows adversarial gradients since they are copies of a legitimate gradient, and the algorithm is only designed for the detection of illegitimate gradients that are not produced through training. Since the algorithm is unable to detect the attacks, the same parameter overflow issue from the victim gradients

4. We provide source code for our attempt at preventing the overflow error in the Krum algorithm at `https://gist.github.com/codymlewis/ce6d70157893068175582fda51097011`.

Environment	Algorithm	Mean ASR (STD ASR)		
		10% Adv.	30% Adv.	50% Adv.
MNIST	FoolsGold	$p_{\mathcal{S}} = 0$	$p_{\mathcal{S}} = 0$	$p_{\mathcal{S}} = 0$
	Krum	0.994 (0.0738)	$p \gg 1$	$p \gg 1$
	STD-DAGMM	$p \gg 1$	$p \gg 1$	$p \gg 1$
	CONTRA	0.335 (0.49)	0.68 (0.378)	$p \gg 1$
	Median	0.988 (0.0154)	0.996 (0.00847)	$p \gg 1$
CIFAR-10	FoolsGold	$p_{\mathcal{S}} = 0$	$p_{\mathcal{S}} = 0$	$p_{\mathcal{S}} = 0$
	Krum	$p \gg 1$	$p \gg 1$	$p \gg 1$
	STD-DAGMM	$p \gg 1$	$p \gg 1$	$p \gg 1$
	CONTRA	0.45 (0.459)	0.252 (0.507)	$p \gg 1$
	Median	0.982 (0.0191)	0.997 (0.00537)	$p \gg 1$
KDD Cup '99	FoolsGold	$p = 0$	$p = 0$	$p = 0$
	Krum	$p \gg 1$	$p \gg 1$	$p \gg 1$
	STD-DAGMM	$p \gg 1$	$p \gg 1$	$p \gg 1$
	CONTRA	0.0 (0.0101)	0.587 (0.385)	$p \gg 1$
	Median	0.957 (0.322)	0.991 (0.0378)	$p \gg 1$

Table 5: Results of the Bad Mouthing Experiments.

as with the Multi-Krum algorithm occurs. This is reflected in the results with $p \gg 1$ due to the complete success of attack.

The impact of the mouthing attacks on the CONTRA algorithm are unique in that they are inverted. The good mouthing attack bad mouths the victim while the bad mouthing attack good mouths the victim. We see that for good mouthing attacks, the influence of the victim's updates have reduced impact on the global update, since CONTRA aims to find malicious collusion between updates, it collects the victim update with the adversary updates and reduces all their impact. The scoring system makes all these updates to maintain a lower impact. At 50% adversaries, the entire group including the victim are determined to be adversaries and are assigned zero impact on the global update. While for the bad mouthing attack, the victim's updates are boosted in influence. Since the adversary updates are the exact opposite of the victim's updates, and that the adversaries' updates exhibit colluding malicious behaviour, they are removed by Contra; this in turn boosts the influence of victim's updates. The impact of this effect is lowered compared to FoolsGold due to the scoring systems reducing the impact of adversary updates. The scoring system has greater impact when there are more adversaries, since more examples are given to punish the behaviour. So bad mouthing has minimal impact at 30% adversaries except in the case of CIFAR-10, where the more complex environment enables a greater good mouthing effect. At 50% adversaries, the bad mouthing attack causes the same progressive overshooting issue seen with Multi-Krum and STD-DAGMM, as to some extent the bad mouthing gradients

are still contributing to the global update, and with this number of adversaries, even a small influence from each is sufficient to derail the learning process.
The Median algorithm is significantly impacted by both attacks. In the case of good mouthing, having more instances of the same gradient makes it closer to the median, while in the case of bad mouthing, having instances of its direct opposite makes the victim gradient to move further from the median. The same progressive overshooting issue is seen when the network is composed of 50% bad mouthing adversaries.

5. New Federated Learning Algorithm

In this section, we propose a new federated learning algorithm, Viceroy, which can mitigate successfully our proposed attacks discussed above as well as other parameter poisoning attacks previously proposed in other research works in the literature. The algorithm is shown in Algorithm 1. Proofs of convergence and adversarial tolerance are provided in Appendix B.

5.1 Design

Observing that the stateful history based mechanism of FoolsGold is reasonably effective against each of the proposed attacks (excluding the ones using on-off toggling), we have designed a new federated learning algorithm that penalizes inconsistent contribution activity. We divide the federated optimization updating mechanism into two factors by which gradients are scaled, namely a reputation mechanism, and the importance of the update. The reputation mechanism functions to penalize only changes in each endpoint's activity represented by contributions, while the importance of update handles the penalization of adversarial activity and other desired objectives. In Algorithm 1, without loss of generality, we define a mitigation function, FedScale(G), which takes as input a gradient or collection of gradients, G, and outputs a weighting, p, to be used in the federated averaging step, it may be defined as any gradient scaling mitigation algorithm/function, e.g. FoolsGold in our experiments. As shown by the studied mitigation algorithms, for FedScale(Δ) which observes the current round's gradients, it effectively detects model poisoning attacks with the downside of being more likely to punish honest updates. Alternatively, for FedScale(H) which observes the history of gradients up to the current round, it reduces the rate of punishing honest updates, with the downside that the presence of the gradient history state creates an attack vector for our proposed on-off attacks to exploit. Our proposed reputation mechanism balances the benefits of these two approaches while minimizing the drawbacks.

5.1.1 Reputation Mechanism

We use the reputation mechanism to aggregate and predict the consistency of an endpoint's updates towards the learning goal. If an endpoint is found to change its learning goal, as in the case of an on-off attack, there are repercussions which are captured using the reputation mechanism. This adds robustness to the stateful historical based systems, where sudden changes in the endpoint's state leads to desynchronization with the server's cached state. Hence, the mechanism improves the reliability (which we define as the degree of synchronization) of the cached state in comparison to most recent one. We calculate the

Algorithm 1 FL algorithm mitigating the proposed attacks

function ViceroY(Set of client gradients Δ_{t+1}, set of client histories H_t, set of client reputations R_t, global model θ_t, history decay factor ω, reputation update factor η)
  $H_{t+1}, R_{t+1} \leftarrow$ ReputationUpdate$(\Delta_{t+1}, H_t, R_t, \omega, \eta)$
  $p \leftarrow R_{t+1} \odot$ FedScale$(H_{t+1}) + (1 - R_{t+1}) \odot$ FedScale(Δ_{t+1})
  $\theta_{t+1} \leftarrow \theta_t + \sum_{i \in \mathcal{U}} p_i \Delta_{t+1,i}$ ▷ Federated averaging aggregation
end function
function ReputationUpdate$(\Delta_{t+1}, H_t, R_t, \omega, \eta)$
  for all $i \in \mathcal{U}$ **do** ▷ Build the history and reputation
    $p_i^H \leftarrow$ FedScale$(H_{t,i})$
    $p_i^c \leftarrow$ FedScale$(\Delta_{t+1,i})$
    $R_{t+1,i} \leftarrow \text{clip}(R_{t,i} + \frac{\eta}{2}(1 - 2|p_i^H - p_i^c|), 0, 1)$
    $H_{t+1,i} \leftarrow \omega H_{t,i} + \Delta_{t+1,i}$
  end for
  return H_{t+1}, R_{t+1}
end function

reputation ($R_{t+1,i}$) at round $t + 1$ for each endpoint, i, as a real number in the range $[0, 1]$ based on the collective disparity between the federated scaling factor of the historical updates ($H_{t,i}$) up to round t and the current update ($\Delta_{t+1,i}$). At each round, the reputation value of the endpoint is incremented with the product of the step size η and the similarity between the scaling factor of the historical updates, and the scaling factor of the current update. To cater for both rewards and punishments, according to the amount of divergence between the scaling factors, the increment of reputation is calculated proportionally to their distance. After increment, the entire reputation value is folded into the range $[-1, 1]$ using the function $\text{clip}(x, a, b) = \max(\min(x, a), b)$.

After computing the reputation, we also aggregate the history. For this aggregation, we decay the historical update by a factor, ω, then add the current update. We decay old observations since, as observed in Section 4.2, the older observations have greater impact on the resulting vector angles. Therefore, through decay we allow the more recent updates to take greater precedence over the older updates.

5.1.2 Update Importance

The importance of endpoints' updates can be determined by any mechanism that mitigates FL Sybil attacks based on their respective update history. For the sake of generality, in Algorithm 1 we name this function FedScale. In our implementation, we have applied a modified FoolsGold, both not to update the history and the global model. Instead, it is a function that has as inputs the endpoint update histories, and outputs the amount by which to scale those respective endpoints' updates. FedScale has the same "modified" functionality. The returned scaling values are then multiplied element-wise, denoted as $\odot$, by a per-endpoint interpolation formed by the reputations. Through this process, smaller reputations cause the current activities of an endpoint to be representative of their trustworthiness, while larger reputations place greater emphasis of the endpoints' historical activities.

5.2 Experiments and Results

In our experiments, we set the parameters $\{\omega, \eta\}$ in our algorithm to $\{0.525, 0.2\}$. The ω parameter was chosen to allow a number of rounds of non-contributions (approximately 56) to decay the historical update vector to $\mathbf{0}$. The η parameter was then selected as it takes 10 completely expected contributions to achieve a reputation of 1. This short turnaround is required as the system overall rewards contributions that are somewhat original within the endpoint's history, this requirement is further observed with η values less than 0.2 which cause the mechanism to require more than 500 completely expected contributions to achieve a reputation of 1. The number of rounds for each of these two factors primarily provide a design criteria for the overall system. For example, under our chosen parameters, the system owner knows that there is the possibility of unfiltered attacks to be present every 56 rounds. Hence, they can checkpoint the system to recover the system to some state prior to the previous 56 rounds if a benchmark fails. We provide equations that find the values of the ω and η parameters respective to the number of rounds corresponding to each functionality in Appendix D.

In Table 5.2, we have shown that the proposed algorithm mitigates successfully the on-off label flipping, on-off free rider, canonical label flipping as well as good mouthing and bad mouthing attacks. It is worth noting that in our on-off attack evaluation, the adversary is assumed to have knowledge of the Viceroy algorithm. In this sense, our proposed mitigation strategy in Viceroy is adaptive. However, despite being effective for any number of adversaries, we have observed a limitation to this defense mechanism, which comes from the underlying Foolsgold (Fung et al., 2020). For the defense mechanism to work, there must exist some honest endpoint(s) which needs to hold the true data of the classes that the adversaries are attempting to manipulate via label flipping or backdoor. We additionally note and provide a solution in Appendix C.4 to the algorithms limited mitigation of single adversaries.

We present mitigation performance against mouthing attacks in Table 5.2, where our algorithm removes only the adversaries, improving the underlying FoolsGold algorithm. Additionally, this helps to clarify the advantage of this algorithm over the Median algorithm, as our algorithm, unlike Median, is not susceptible to the manipulations of per-endpoint element distributions caused by attacks such as scaled poisoning, or the mouthing variants proposed in this paper.

For the sake of comparison, in Tables 5.2 and 5.2, we have repeated the results of the previously studied algorithms for each respective attack learning on the MNIST data set.[5]

Our algorithm does not resolve the issues that emerge from label flipping on class imbalanced data, as seen from the negative results for both label flipping variants on the KDD Cup '99 data set. This presents a future research direction to our work to consider substitution of the Federated SGD update applied in the FLDS algorithms with data heterogeneity such as in FedMAX (Chen et al., 2020) and FedProx (Li et al., 2018).

5. Results for label flipping on the other federated learning defense systems that we have studied are omitted since the attacks are effectively mitigated, with the caveats presented in their respective original proposals.

Attack	Environment	Mean ASR (STD ASR)		
		10% Adv.	30% Adv.	50% Adv.
On-off Label Flip	MNIST	2.86e-5 (3e-4)	1e-4 (1.2e-4)	0.135 (0.316)
	CIFAR-10	1.23e-4 (2.2e-4)	7.6e-4 (3e-4)	0.185 (0.358)
	KDD Cup '99	0.999 (2.23e-3)	0.998 (0.0446)	0.995 (0.064)
	FoolsGold	0.241 (0.314)	0.725 (0.445)	0.993 (0.0828)
	Krum	7.7e-4 (4.8e-4)	4.87e-4 (0.0158)	5.9e-5 (3.75e-3)
	STD-DAGMM	0.0253 (0.0956)	0.889 (0.126)	0.988 (0.0169)
	CONTRA	0.199 (0.31)	0.458 (0.441)	0.539 (0.481)
	Median	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)
On-off Free ride	MNIST	0.378 (0.485)	0.256 (0.436)	0.002 (0.0447)
	CIFAR-10	0.249 (0.455)	0.196 (0.316)	0.0254 (0.0566)
	KDD Cup '99	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)
	FoolsGold	0.982 (0.133)	0.243 (0.429)	0.0 (0.0)
	Krum	0.448 (0.454)	0.35 (0.355)	0.25 (0.253)
	STD-DAGMM	1.0 (0.0)	1.0 (0.0)	0.5 (0.5)
	CONTRA	0.308 (0.452)	0.383 (0.431)	0.403 (0.433)
	Median	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)
Label Flipping	MNIST	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)
	CIFAR-10	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)
	KDD Cup '99	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)
Scaled Backdoor	MNIST	0.15 (0.353)	0.2 (0.393)	0.215 (0.4)
	CIFAR-10	0.0872 (0.248)	0.09 (0.378)	0.159 (0.457)
	KDD Cup '99	0.994 (0.0772)	0.979 (0.143)	0.998 (0.0447)
	FoolsGold	0.827 (0.378)	0.989 (0.106)	1.0 (0.0)
	Krum	1.3e-3 (0.0342)	1.2e-5 (1.25e-4)	2e-6 (9.1e-5)
	STD-DAGMM	0.987 (0.0449)	1.0 (6.8e-5)	1.0 (0.0)
	CONTRA	1.0 (9.25e-4)	1.0 (0.0127)	0.999 (0.0237)
	Median	0.00368 (0.0124)	0.126 (0.222)	1.0 (0.0125)

Table 6: Experimental results of Viceroy with on-off and canonical attacks. The Environment column consists of the Viceroy algorithm experiments on the three data sets, MNIST, CIFAR-10, and KDD Cup '99, and our previous results from the studied federated learning defense systems on the MNIST data set.

Attack	Environment	Mean ASR (STD ASR)		
		10% Adv.	30% Adv.	50% Adv.
Good Mouthing	MNIST	$p_{\mathcal{S}} = 0$	$p_{\mathcal{S}} = 0$	$p_{\mathcal{S}} = 0$
	CIFAR-10	$p_{\mathcal{S}} = 0$	$p_{\mathcal{S}} = 0$	$p_{\mathcal{S}} = 0$
	KDD Cup '99	$p = 0$	$p = 0$	$p = 0$
	FoolsGold	$p_{\mathcal{S}} = 0$	$p_{\mathcal{S}} = 0$	$p_{\mathcal{S}} = 0$
	Krum	$p \gg 1$	$p \gg 1$	$p \gg 1$
	STD-DAGMM	$p \gg 1$	$p \gg 1$	$p \gg 1$
	CONTRA	0.15 (0.821)	0.216 (0.8)	$p = 0$
	Median	0.968 (0.027)	0.983 (0.0245)	0.986 (0.0222)
Bad Mouthing	MNIST	$p_{\mathcal{S}} = 0$	$p_{\mathcal{S}} = 0$	$p_{\mathcal{S}} = 0$
	CIFAR-10	$p_{\mathcal{S}} = 0$	$p_{\mathcal{S}} = 0$	$p_{\mathcal{S}} = 0$
	KDD Cup '99	$p = 0$	$p = 0$	$p = 0$
	FoolsGold	$p_{\mathcal{S}} = 0$	$p_{\mathcal{S}} = 0$	$p_{\mathcal{S}} = 0$
	Krum	0.994 (0.0738)	$p \gg 1$	$p \gg 1$
	STD-DAGMM	$p \gg 1$	$p \gg 1$	$p \gg 1$
	CONTRA	0.335 (0.49)	0.68 (0.378)	$p \gg 1$
	Median	0.988 (0.0154)	0.996 (0.00847)	$p \gg 1$

Table 7: Results of the Viceroy Experiments with mouthing attacks. The Environment column consists of the Viceroy algorithm experiments on the three data sets, MNIST, CIFAR-10, and KDD Cup '99, and our previous results from the studied federated learning defense systems on the MNIST data set.

6. Conclusion

In this paper, we have shown that federated learning defense systems are susceptible to our proposed attacks, which include on-off label flipping, good/bad mouthing, and on-off free riding attacks. In the on-off attacks, the adversary evades the mechanisms and penalties imposed by the defense systems by alternating between attacking and acting normal. With good and bad mouthing attacks, adversaries send selected gradients that either enhance or reduce the impact of the chosen victim's gradients on the global model. We have analysed each of these attacks using multiple data sets and shown that they are successful against existing defense systems in federated learning. We have implemented all these attacks on five different FL algorithms using three different data sets, and two neural network models. These results show that the proposed attacks are effective in all these cases. Then we have developed a new federated learning algorithm which we have shown can mitigate each of the proposed attacks simultaneously, while remaining effective against previously proposed attacks.

Acknowledgments

This research is supported by an Australian Government Research Training Program (RTP) Scholarship.

Appendix Appendix A. Proof of the Model Direction Equation

Consider the federated optimization function given by Equation (3), where for each endpoint's locally held model, $\psi_{t+1,u}$, $\Delta_{t+1,u} = \theta_t - \psi_{t+1,u}$ is the gradient sent by endpoint u for time $t+1$ and $n_{t+1,u}$ is u's chosen learning rate for time $t+1$.

$$\theta_{t+1} = \theta_t - \sum_{u \in \mathcal{U}} p_{t+1,u} n_{t+1,u} (\theta_t - \psi_{t+1,u}) \tag{3}$$

Proof Let $P(\tau) = \theta_{t+\tau}$. For the base case $\tau = 1$,

$$\begin{aligned} P(1) &= \theta_t (1 - \sum_{u \in \mathcal{U}} n_{t+1,u} p_{t+1,u}) + \sum_{u \in \mathcal{U}} \psi_{t+1,u} n_{t+1,u} p_{t+1,u} \\ &= \theta_t - \sum_{u \in \mathcal{U}} n_{t+1,u} p_{t+1,u} (\theta_t - \psi_{t+1,u}), \end{aligned}$$

which is equal to Equation (3), therefore the statement holds true for $P(1)$.

Next, assume $P(k) \to T$,

$$P(k) = \theta_t \prod_{i=1}^{k} \left(1 - \sum_{u \in \mathcal{U}} n_{t+i,u} p_{t+i,u} \right) + \sum_{i=1}^{k} \sum_{u \in \mathcal{U}} \psi_{t+i,u} n_{t+i,u} p_{t+i,u} \prod_{j=1}^{k-i} (1 - n_{t+j+1,u} p_{t+j+1,u}).$$

Induce for $P(k+1)$,

$$\begin{aligned} P(k+1) &= \theta_{t+k+1} \\ &= \theta_t \prod_{i=1}^{k+1} \left(1 - \sum_{u \in \mathcal{U}} n_{t+i,u} p_{t+i,u} \right) \\ &\quad + \sum_{i=1}^{k+1} \sum_{u \in \mathcal{U}} \psi_{t+i,u} n_{t+i,u} p_{t+i,u} \prod_{j=1}^{k+1-i} (1 - n_{t+j+1,u} p_{t+j+1,u}). \end{aligned}$$

This gives a right hand side of,

$$
\begin{aligned}
RHS = (1 - \sum_{u \in \mathcal{U}} n_{t+k+1,u} p_{t+k+1,u}) \theta_t \prod_{i=1}^{k+1} \left(1 - \sum_{u \in \mathcal{U}} n_{t+i,u} p_{t+i,u} \right) \\
+ \sum_{u \in \mathcal{U}} \psi_{t+k+1,u} n_{t+k+1,u} p_{t+k+1,u} \\
+ (1 - \sum_{u \in \mathcal{U}} n_{t+k+1,u} p_{t+k+1,u}) \sum_{i=1}^{k} \sum_{u \in \mathcal{U}} \psi_{t+i,u} n_{t+i,u} p_{t+i,u} \prod_{j=1}^{k-i} (1 - n_{t+j+1,u} p_{t+j+1,u}) \\
= (1 - \sum_{u \in \mathcal{U}} n_{t+k+1,u} p_{t+k+1,u}) (\theta_t \prod_{j=1}^{k} (1 - \sum_{u \in \mathcal{U}} n_{t+i,u} p_{t+i,u}) \\
+ \sum_{i=1}^{k} \sum_{u \in \mathcal{U}} \psi_{t+i,u} n_{t+i,u} p_{t+i,u} \prod_{j=1}^{k-i} (1 - n_{t+j+1,u} p_{t+j+1,u})) \\
+ \sum_{u \in \mathcal{U}} n_{t+k+1,u} p_{t+k+1,u} \psi_{t+k+1,u} \\
= (1 - \sum_{u \in \mathcal{U}} n_{t+k+1,u} p_{t+k+1,u}) \theta_{t+k} + \sum_{u \in \mathcal{U}} n_{t+k+1,u} p_{t+k+1,u} \psi_{t+k+1,u}.
\end{aligned}
$$

And by expanding the left hand side according to Equation (3),

$$
\begin{aligned}
LHS &= (1 - \sum_{u \in \mathcal{U}} n_{t+k+1,u} p_{t+k+1,u}) \theta_{t+k} + \sum_{u \in \mathcal{U}} n_{t+k+1,u} p_{t+k+1,u} \psi_{t+k+1,u} \\
&= RHS,
\end{aligned}
$$

the statement holds true for $P(k+1)$. Therefore this equation holds with respect to $\forall \tau \in \mathbb{N}$. ■

Appendix Appendix B. Proof of Viceroy Convergence

For the proofs presented in this section, we assume that the mini-batch gradients are unbiased estimators of the batch gradient. We therefore drop the respective notations of stochastic gradient descent in favor of gradient descent, due to equivalence in this case. These proofs could be trivially extended for cases where the mini-batch gradients provide biased estimates, which would simply add estimate variance terms to the resulting bounds.

Appendix B.1 Within an Honest and Consistent network

Assume for the following update rules,

$$
\theta^H = \theta - \sum_{i \in \mathcal{U}} \textsc{FedScale}(H_i) n \nabla f_i(\theta) = \theta - \frac{1}{|\mathcal{U}|} \sum_{i \in \mathcal{U}} n \nabla f_i(\theta) \tag{4}
$$

$$\theta^d = \theta - \sum_{i \in \mathcal{U}} \textsc{FedScale}(\Delta_i) n \nabla f_i(\theta) = \theta - \frac{1}{|\mathcal{U}|} \sum_{i \in \mathcal{U}} n \nabla f_i(\theta). \tag{5}$$

Then the reputation scaled average update of (4) and (5) can be expressed as follows,

$$\begin{aligned}\theta^R &= \theta - \sum_{i \in \mathcal{U}} (R_i \textsc{FedScale}(H_i) + (1 - R_i) \textsc{FedScale}(\Delta_i)) n \nabla f_i(\theta) \\ &= \theta - \sum_{i \in \mathcal{U}} (R_i \frac{1}{|\mathcal{U}|} + (1 - R_i) \frac{1}{|\mathcal{U}|}) n \nabla f_i(\theta),\end{aligned}$$

given that all endpoints are honest and consistent with updates, all R_i are equal. Defining $r = R_i$,

$$\theta^R = \theta - \sum_{i \in \mathcal{U}} (r \frac{1}{|\mathcal{U}|} + (1 - r) \frac{1}{|\mathcal{U}|}) n \nabla f_i(\theta) = \theta - \sum_{i \in \mathcal{U}} \frac{1}{|\mathcal{U}|} n \nabla f_i(\theta) \tag{6}$$

This is equivalent to the federated GD update,

$$\begin{aligned}\theta^+ &= \theta - n \mathbb{E}[\nabla f_i(\theta)] \\ &= \theta - \frac{1}{|\mathcal{U}|} \sum_{i \in \mathcal{U}} n \nabla f_i(\theta).\end{aligned}$$

Lemma 1 *The federated gradient descent update is equivalent to a gradient descent update on the union of endpoint's data sets when all data sets are of equal size. That is,*

$$\theta - \frac{1}{|\mathcal{U}|} \sum_{i \in \mathcal{U}} n \nabla f_i(\theta) = \theta - n \nabla f(\theta).$$

Proof Define the data sets as $D = \bigcup_{i \in \mathcal{U}} D_i$, and the gradient of the objective function as,

$$\nabla f(\theta) = \nabla \frac{1}{|D|} \sum_{d \in D} l(d, \theta),$$

where $l(d, \theta)$ is the loss function evaluated on the model's prediction of sample d.
Then the federated gradient descent can be expanded as follows,

$$\frac{1}{|\mathcal{U}|} \sum_{i \in \mathcal{U}} \nabla f_i(\theta) = \frac{1}{|\mathcal{U}|} \sum_{i \in \mathcal{U}} \nabla \frac{1}{|D_i|} \sum_{d \in D_i} l(d, \theta). \tag{7}$$

Since all data sets are of equal size, $|D| = \sum_{i \in \mathcal{U}} |D_i|$ implies that $|D_i| = \frac{|D|}{|\mathcal{U}|}$, so (7) can be simplified as follows,

$$\begin{aligned}\frac{1}{|\mathcal{U}|}\sum_{i\in\mathcal{U}}\nabla\frac{1}{|D_i|}\sum_{d\in D_i}l(d,\theta) &= \frac{1}{|\mathcal{U}|}\sum_{i\in\mathcal{U}}\nabla\frac{|\mathcal{U}|}{|D|}\sum_{d\in D_i}l(d,\theta)\\ &= \sum_{i\in\mathcal{U}}\nabla\frac{1}{|D|}\sum_{d\in D_i}l(d,\theta)\\ &= \nabla\frac{1}{|D|}\sum_{d\in D}l(d,\theta).\end{aligned}$$

The last line follows from the fact that $D=\bigcup_{i\in\mathcal{U}}D_i$. ■

Using Lemma 1, the proof of convergence is equivalent to the proof of gradient descent convergence as follows.

Proof Assume that the objective function, $f:\mathbb{R}^n\mapsto\mathbb{R}$, is continuously differentiable and Lipschitz smooth,

$$f(y)\leq f(x)+\langle\nabla f(x),y-x\rangle+\frac{L}{2}\|y-x\|^2. \tag{8}$$

Then the expansion of (8) evaluated at the current model, θ, and centered at the gradient descent update rule, $\theta^+=\theta-n\nabla f(\theta)$, is as follows,

$$\begin{aligned}f(\theta^+) &\leq f(\theta)+\langle\nabla f(\theta),\theta^+-\theta\rangle+\frac{L}{2}\|\theta^+-\theta\|^2\\ &= f(\theta)+\langle\nabla f(\theta),\theta-n\nabla f(\theta)-\theta\rangle+\frac{L}{2}\|\theta-n\nabla f(\theta)-\theta\|^2\\ &= f(\theta)+\langle\nabla f(\theta),-n\nabla f(\theta)\rangle+\frac{L}{2}\|-n\nabla f(\theta)\|^2\\ &= f(\theta)-n\nabla f(\theta)^2+\frac{L}{2}n^2\nabla f(\theta)^2\\ &= f(\theta)-n\nabla f(\theta)^2(1-\frac{L}{2}n).\end{aligned}$$

Setting $n\leq\frac{1}{L}$ which implies $(1-\frac{L}{2}n)\leq\frac{1}{2}$,

$$f(\theta^+)\leq f(\theta)-\frac{n}{2}\nabla f(\theta)^2.$$

Evaluating the optimal model $\theta^*=\arg\min_\theta f(\theta)$ with the differential convex function property,

$$\begin{aligned}f(\theta^*)&\geq f(\theta)+\langle\nabla f(\theta^*),\theta^*-\theta\rangle\\ -f(\theta^*)&\leq\langle\nabla f(\theta^*),\theta-\theta^*\rangle-f(\theta).\end{aligned}$$

This allows us to form the following inequality by looking at the difference between the objective function at gradient descent update and optimal model parameters,

$$f(\theta^+) - f(\theta^*) \leq \frac{1}{2n}(2n\langle \nabla f(\theta^*), \theta - \theta^* \rangle - n^2 \nabla f(x)^2$$
$$= \frac{1}{2n}(\|\theta - \theta^*\|^2 - \|\theta^+ - \theta^*\|^2).$$

Expanding this inequality into a sum of the updates up to round $\mathcal{T}$,

$$\sum_{i=1}^{\mathcal{T}} f(\theta^{(i)}) - f(\theta^*) \leq \sum_{i=1}^{\mathcal{T}} \frac{1}{2n}(\|\theta^{(i-1)} - \theta^*\|^2 - \|\theta^{(i)} - \theta^*\|^2)$$
$$= \frac{1}{2n}(\|\theta^{(0)} - \theta^*\|^2 - \|\theta^{(\mathcal{T})} - \theta^*\|^2)$$
$$\leq \frac{1}{2n}\|\theta^{(0)} - \theta^*\|^2.$$

We can therefore retrieve the upper bound for the distance between the current round's objective function value and the optimal through expectation as follows,

$$f(\theta^{(\mathcal{T})}) - f(\theta^*) \leq \frac{1}{\mathcal{T}} \sum_{i=1}^{\mathcal{T}} (f(\theta^{(i)}) - f(\theta^*))$$
$$\leq \frac{1}{2\mathcal{T}n}\|\theta^{(0)} - \theta^*\|^2.$$

■

Appendix B.2 Convergence within Adversarial Environments

We first assume that attacks may be mitigated by both scaling mechanisms, when adversaries do not alter their attack patterns. The following proof shows that our proposed algorithm converges within this environment.

Proof With the sets of honest endpoints, $\mathcal{H}$, and adversarial endpoints, $\mathcal{S}$, we redefine the data set to optimize over as the union of honest endpoint data sets, $D = \bigcup_{i \in \mathcal{H}} D_i$.

We then define the scaling function from (5) as,

$$\textsc{FedScale}(\Delta_i) = \begin{cases} \frac{1}{|\mathcal{H}|}, & i \in \mathcal{H} \\ 0, & \text{otherwise} \end{cases}, \tag{9}$$

hence the proof of convergence from Appendix B.1 holds for $r = 0$.

Additionally, we define scaling function from (4) for completely consistent updates as,

$$\textsc{FedScale}(H_i) = \begin{cases} \frac{1}{|\mathcal{H}|}, & i \in \mathcal{H} \\ 0, & \text{otherwise} \end{cases}, \tag{10}$$

allowing the proof of convergence from Appendix B.1 to hold for $r = 1$. ■

To prove that our proposed algorithm converges when adversaries toggle between submitting honest and adversarial updates, we first prove three lemmas which respectively find the exact bound for the complete change of reputation value, the upper bound for the convergence of the history towards an adversarial state, and the lower bound for the convergence of model parameters into the adversaries' target state. We then use these bounds to show that with careful selection of the respective hyper-parameters, the proposed reputation and history mechanisms effectively mitigate on-off model poisoning attacks.

Lemma 2 *The reputation mechanism converges to its extreme values of 0 or 1 with rate* $\mathcal{T} = \frac{2}{\eta}$ *when there is a consistent complete agreement or disagreement between weightings found by the stateful and stateless systems.*

Proof For $p^H = \textsc{FedScale}(H)$ and $p^c = \textsc{FedScale}(\Delta)$,

$$R_t = \begin{cases} R_{t-1} + \frac{\eta}{2}(1 - 2|p^H - p^c|) & : t > 0 \\ 0 & : otherwise \end{cases}. \tag{11}$$

Assume that if the updates are completely consistent with the history, $|p^H - p^c| = 0$ and $|p^H - p^c| = 1$ otherwise. We have,

$$R_{t+1} = R_t + \frac{\eta}{2}$$
$$R_{\mathcal{T}} = \sum_{i=1}^{\mathcal{T}} \frac{\eta}{2}.$$

Then the time until convergence of reputation from 0 to 1 is $\mathcal{T} = \frac{2}{\eta}$.

In the other extreme where the updates are completely inconsistent with the history, $|p^H - p^c| = 1$,

$$R_{t+1} = R_t - \frac{\eta}{2}$$
$$R_{\mathcal{T}} = -\sum_{i=1}^{\mathcal{T}} \frac{\eta}{2}.$$

Then the time until convergence of reputation from 1 to 0 is $\mathcal{T} = \frac{2}{\eta}$. ■

Lemma 3 *The number of rounds from round* t *to converge to a history recognised by the system as adversarial is upper bounded by* (12) *when adversarial updates are continually submitted.*

$$\mathcal{T} < \log_\omega(\epsilon) - \log_\omega(\|H_t\|) \tag{12}$$

Proof Define the adversarial history $H_a = \sum_{i=0}^{\mathcal{T}-1} \omega^i \Delta_a$, and the endpoint's history from round t up to round $t+\mathcal{T}$ as $\omega^{\mathcal{T}} H_t + \sum_{i=0}^{\mathcal{T}-1} \omega^i \Delta_a$. Then as follows, by solving the inequality formed by the distance between the adversarial history and the endpoints' history being bounded by the hyper-parameter ϵ which states the maximum distance between parameters before their behaviour becomes sufficiently equivalent,

$$\begin{aligned} \|\omega^{\mathcal{T}} H_t + \sum_{i=0}^{\mathcal{T}-1} (\omega^i \Delta_a) - H_a\| &< \epsilon \\ \|\omega^{\mathcal{T}} H_t\| &< \epsilon \\ \omega^{\mathcal{T}} &< \frac{\epsilon}{\|H_t\|} \\ \mathcal{T} &< \log_\omega(\epsilon) - \log_\omega(\|H_t\|). \end{aligned}$$

Directly proving the upper bound for the history's convergence. ■

Lemma 4 *The distance between adversarial objective functions $g : \mathbb{R}^n \mapsto \mathbb{R}$ with respect to the model after $\mathcal{T}$ rounds of model poisoning updates, $\theta^{(\mathcal{T})}$, and the adversary's target model, $\theta^a = \arg\min_\theta g(\theta)$,is lower bounded by* (13)*.*

$$g(\theta^{(\mathcal{T})} - g(\theta^a) \geq \frac{1}{n\mathcal{T}} (\|\theta^{(0)} - \theta^a\|^2 - \|\theta^{(\mathcal{T})} - \theta^a\|^2) \tag{13}$$

Proof In the following, we assume that in terms of the honest objective function, the optimal model and the adversarial target model are functionally distant as discussed by the no free lunch theorem (Papernot et al., 2016), $f(\theta^a) - f(\theta^*) > \delta$, and use $\nabla g_i(\theta)$ to denote the gradient that directs the model towards θ^a.

In a federated learning system with no defenses against model poisoning, the update will be as follows, where $\nabla f(\theta) = \frac{1}{|\mathcal{H}|} \sum_{i \in \mathcal{H}} \nabla f_i(\theta)$ and $\nabla g(\theta) = \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} \nabla g_i(\theta)$,

$$\theta^+ = \theta - n \left(\frac{|\mathcal{H}|}{|\mathcal{U}|} \nabla f(\theta) + \frac{|\mathcal{S}|}{|\mathcal{U}|} \nabla g(\theta) \right).$$

For this proof, we assume that the adversarial objective function is continuously differentiable, Lipschitz smooth, and satisfies μ-strong convexity, illustrated by the inequality (14).

$$g(y) \geq g(x) + \langle \nabla g(x), y - x \rangle + \frac{\mu}{2} \|y - x\|^2 \tag{14}$$

We can then find the lower bound for the number of rounds required for convergence of the model towards the adversary's target. First we will substitute the model update and current model into (14), resulting in the following. Where to better fit the equations within the borders we sparingly use m to denote $\frac{\mu n^2}{2|\mathcal{U}|^2}$ and $\nabla fg(\theta)$ to denote $\nabla f(\theta) \nabla g(\theta)$,

$$
\begin{aligned}
g(\theta^+) &\geq g(\theta) + \langle \nabla g(\theta), \theta^+ - \theta \rangle + \frac{\mu}{2}\|\theta^+ - \theta\|^2 \\
&= g(\theta) + \langle \nabla g(\theta), -\frac{n}{|\mathcal{U}|}\left(|\mathcal{H}|\nabla f(\theta) + |\mathcal{S}|\nabla g(\theta)\right)\rangle + \frac{\mu n^2}{2|\mathcal{U}|^2}\||\mathcal{S}|\nabla g(\theta) + |\mathcal{H}|\nabla f(\theta)\|^2 \\
&= g(\theta) - \frac{n|\mathcal{S}|}{|\mathcal{U}|}\nabla g(\theta)^2 - \frac{n|\mathcal{H}|}{|\mathcal{U}|}\nabla f g(\theta) + m\||\mathcal{S}|^2\nabla g(\theta)^2 + 2|\mathcal{H}||\mathcal{S}|\nabla f g(\theta) + |\mathcal{H}|^2\nabla f(\theta)^2\| \\
&= g(\theta) + \frac{n|\mathcal{S}|}{|\mathcal{U}|}\nabla g(\theta)^2\left(\frac{\mu n|\mathcal{S}|}{2|\mathcal{U}|} - 1\right) + \frac{n|\mathcal{H}|}{|\mathcal{U}|}\nabla f(\theta)\nabla g(\theta)\left(\frac{|\mathcal{S}|\mu n}{|\mathcal{U}|} - 1\right) + |\mathcal{H}|^2 m\nabla f(\theta)^2.
\end{aligned}
$$

Set $n \geq \frac{1}{\mu}$ which means $\frac{\mu n a}{c} - 1 \geq -\frac{c-a}{c}$ allowing us to simplify the inequality,

$$
\begin{aligned}
g(\theta^+) &\geq g(\theta) - \frac{n|\mathcal{S}|}{|\mathcal{U}|}\nabla g(\theta)^2\frac{2|\mathcal{U}| - |\mathcal{S}|}{2|\mathcal{U}|} - \frac{n|\mathcal{H}|}{|\mathcal{U}|}\nabla f(\theta)\nabla g(\theta)\frac{|\mathcal{U}| - |\mathcal{S}|}{|\mathcal{U}|} + \frac{|\mathcal{H}|^2 n}{2|\mathcal{U}|^2}\nabla f(\theta)^2 \\
&\geq g(\theta) - n\frac{|\mathcal{S}|(2|\mathcal{U}| - |\mathcal{S}|)}{2|\mathcal{U}|^2}\nabla g(\theta)^2 - n\frac{|\mathcal{H}|(|\mathcal{U}| - |\mathcal{S}|)}{|\mathcal{U}|^2}\nabla f(\theta)\nabla g(\theta) - n\frac{|\mathcal{H}|^2}{|\mathcal{U}|^2}\nabla f(\theta)^2,
\end{aligned}
$$

where the placement of the last term is allowed to occur due to the positive and positive semidefinite properties of its components.

Applying (14) to the current and adversarial target model, we get the following,

$$
\begin{aligned}
g(\theta) &\geq g(\theta^a) + \langle \nabla g(\theta^a), \theta - \theta^a \rangle + \frac{\mu}{2}\|\theta - \theta^a\|^2 \\
&= g(\theta^a) + \frac{\mu}{2}\|\theta - \theta^a\|^2 \\
-g(\theta^a) &\geq \frac{\mu}{2}\|\theta - \theta^a\|^2 - g(\theta).
\end{aligned}
$$

Allowing us to take the distance between the objective functions evaluated at the updated model and the adversarial target model. Where $-\frac{|\mathcal{S}|(2|\mathcal{U}|-|\mathcal{S}|)}{2|\mathcal{U}|^2} \geq -\frac{|\mathcal{S}|^2}{|\mathcal{U}|^2}$ and $-\frac{|\mathcal{H}|(|\mathcal{U}|-|\mathcal{S}|)}{|\mathcal{U}|^2} \geq -\frac{|\mathcal{H}|^2}{|\mathcal{U}|^2}$, and we assume that there are less or equal adversaries than honest users, $|\mathcal{S}| \leq |\mathcal{H}|$,

$$
\begin{aligned}
g(\theta^+) - g(\theta^a) &\geq \frac{\mu}{2}\|\theta - \theta^a\|^2 - n\frac{|\mathcal{S}|^2}{|\mathcal{U}|^2}\nabla g(\theta)^2 - n\frac{|\mathcal{H}|^2}{|\mathcal{U}|^2}\nabla f(\theta)\nabla g(\theta) - n\frac{|\mathcal{H}|^2}{|\mathcal{U}|^2}\nabla f(\theta)^2 \\
&\geq \frac{1}{n}\left(\frac{1}{2}\|\theta - \theta^a\|^2 - \frac{n^2}{|\mathcal{U}|^2}(|\mathcal{S}|^2\nabla g(\theta)^2 + |\mathcal{S}|^2\nabla f(\theta)\nabla g(\theta) + |\mathcal{S}|^2\nabla f(\theta)^2)\right).
\end{aligned}
$$

Using the following observation, with $\Delta = \frac{|\mathcal{H}|}{|\mathcal{U}|}\nabla f(\theta) + \frac{|\mathcal{S}|}{|\mathcal{U}|}\nabla g(\theta)$,

$$\begin{aligned}
\|\theta^+ - \theta^a\|^2 &= \|\theta - n(\frac{|\mathcal{H}|}{|\mathcal{U}|}\nabla f(\theta) + \frac{|\mathcal{S}|}{|\mathcal{U}|}\nabla g(\theta)) - \theta^*\|^2 \\
&= \|\theta - \theta^*\|^2 - 2n\langle \Delta, \theta - \theta^a\rangle + \frac{n^2}{|\mathcal{U}|^2}(|\mathcal{H}|^2\nabla f(x)^2 + 2|\mathcal{H}||\mathcal{S}|\nabla f g(\theta) + |\mathcal{S}|^2\nabla g(\theta)) \\
&\leq \|\theta - \theta^a\|^2 + \frac{n^2}{|\mathcal{U}|^2}\left(|\mathcal{H}|^2\nabla f(x)^2 + 2|\mathcal{H}||\mathcal{S}|\nabla f(\theta)\nabla g(\theta) + |\mathcal{S}|^2\nabla g(\theta)\right) \\
&\leq \|\theta - \theta^a\|^2 + \frac{n^2}{|\mathcal{U}|^2}\left(|\mathcal{S}|^2\nabla f(x)^2 + |\mathcal{S}|^2\nabla f(\theta)\nabla g(\theta) + |\mathcal{S}|^2\nabla g(\theta)\right),
\end{aligned}$$

we can simplify the distances between the objective functions,

$$\begin{aligned}
g(\theta^+) - g(\theta^a) &\geq \frac{1}{n}\left(\frac{1}{2}\|\theta - \theta^a\|^2 - \|\theta^+ - \theta^a\|^2 + \|\theta - \theta^a\|^2\right) \\
&\geq \frac{1}{n}(\|\theta - \theta^a\|^2 - \|\theta^+ - \theta^a\|^2).
\end{aligned}$$

Allowing the distance at round $\mathcal{T}$ to be lower bounded by the expectation,

$$\begin{aligned}
g(\theta^{(\mathcal{T})}) - g(\theta^a) &\geq \frac{1}{\mathcal{T}}\sum_{i=0}^{\mathcal{T}} g(\theta^{(i)}) - g(\theta^a) \\
&= \frac{1}{n\mathcal{T}}\sum_{i=0}^{\mathcal{T}}(\|\theta^{(i-1)} - \theta^a\|^2 - \|\theta^{(i)} - \theta^a\|^2) \\
&= \frac{1}{n\mathcal{T}}(\|\theta^{(0)} - \theta^a\|^2 - \|\theta^{(\mathcal{T})} - \theta^a\|^2).
\end{aligned}$$

■

Hence, if we bound our proposed algorithm's hyperparameters η and ω by ensuring that the results from Lemmas 2 and 3 are less than the result from Lemma 4, denoted as ϵ, the algorithm will find suitable reputation and history values for mitigation of the on-off toggling attack prior to its impact. The precise bounds for the hyperparameters are presented in (15) and (16),

$$\eta < \frac{2\epsilon n}{\|\theta^{(0)} - \theta^a\|^2 - \|\theta^{(\mathcal{T})} - \theta^a\|^2} \tag{15}$$

$$\omega > \sqrt[b]{\frac{\epsilon}{\|H_t\|}} : b = \frac{1}{\mathcal{T}n}(\|\theta^{(0)} - \theta^a\|^2 - \|\theta^{(\mathcal{T})} - \theta^a\|^2). \tag{16}$$

This concludes the proof.

Environment	Algorithm	Mean ASR (STD ASR)		
		10% Adv.	30% Adv.	50% Adv.
Non-i.i.d. MNIST	FoolsGold	0.485 (0.256)	0.555 (0.405)	0.968 (0.117)
	Krum	0.98 (0.0469)	0.994 (0.0446)	0.997 (0.0446)
	STD-DAGMM	0.0894 (0.246)	0.926 (0.0552)	0.954 (0.0422)
i.i.d. MNIST	FoolsGold	0.797 (0.181)	0.0158 (0.0696)	0.771 (0.331)
	Krum	8.42e-3 (0.0226)	8.58e-3 (0.0265)	0.979 (0.0636)
CIFAR-10	FoolsGold	0.0267 (0.0268)	0.495 (0.174)	0.818 (0.155)
	Krum	0.574 (0.213)	0.95 (0.0582)	0.989 (0.0463)
KDD Cup '99	FoolsGold	0.978 (0.143)	0.98 (0.139)	0.994 (0.0772)
	Krum	0.998 (0.0446)	0.998 (0.0446)	0.998 (0.0446)
$8 \rightarrow 2$, MNIST	FoolsGold	0.0195 (0.0538)	0.0651 (0.1803)	0.524 (0.4)
	Krum	0.961 (0.0484)	0.985 (0.0425)	0.997 (0.0419)
Label Flipping	FoolsGold	0.363 (0.274)	0 (0)	0 (0)
	Krum	0.982 (0.0454)	0.993 (0.0446)	0.997 (0.0446)
	STD-DAGMM	0.898 (0.0548)	0.97 (0.0419)	0.983 (0.0416)
Scaled Backdoor	FoolsGold	0.494 (0.266)	0 (0)	0 (0)
	Krum	0.982 (0.0455)	0.993 (0.0445)	0.997 (0.0446)

Table 8: Results of the On-Off Label Flipping Experiments

Appendix Appendix C. Experiments with Alternative Data Distributions

For our experiments with 10 endpoints, we looked at two ways of distributing the MNIST data across the endpoints, namely, a completely non-i.i.d. distribution and a completely i.i.d. distribution. In non-i.i.d. distribution each endpoint exclusively holds data on one of the classes whereas in the i.i.d. distribution, each endpoint is assigned a class balanced and equally sized partition of the data set. We primarily evaluate under the non-i.i.d. condition, as it is a tougher environment for Sybils in the FL defense systems. This arises from the fact that each honest gradient is equally unique from each other, and Sybils, which produce very similar gradients, become evident as an anomaly (Fung et al., 2020). For the CIFAR-10 data set, we distributed the data in a non-i.i.d. manner with one class per endpoint so that we could evaluate the impact of a more complex learning task on the effectiveness of the attack.

Appendix C.1 On-off Label Flipping

In Table Appendix C.1, we present all the results from the experiments with the on-off label flipping attacks. In the non-i.i.d. MNIST experiments, each endpoint holds the data of a single class. In this environment, the FoolsGold experiments demonstrate the clear trend between the amount of adversaries and the success of the attack. Figure 1 demonstrates an

example of the attack's effects on the overall system run-time with a network of 50% adversaries in a FoolsGold system. The Multi-Krum experiments demonstrate the ineffectiveness of the algorithm in detecting malicious gradients under completely non-i.i.d. data environments. This effect arises from the Krum mechanism filtering out gradients that are furthest from the others. Under a non-i.i.d. data environment, each of the honest gradients are very distant from each other due to learning a unique class. However, malicious gradients are close to each other due to collusion and are therefore selected over the honest gradients. STD-DAGMM fails to mitigate this attack as it is designed only to detect whether a gradient is artificial. So the failure emerges from the poisoned gradients being produced by the adversaries still being generated through the training process as with honest endpoints. The difference being, when in the on state the adversaries train their model with poisoned data. There is a notable exception in the case where there is a single adversary. The resulting ASR is not significant, as in this case STD-DAGMM is able to differentiate the gradients resulting from the adversary's local model over the other honest ones. This is not the case when there are multiple adversaries, as the STD-DAGMM must observe the gradients in lieu of the model that generated them due to each adversary holding their own model. Hence, STD-DAGMM only observes those legitimately generated gradients. Since we have shown STD-DAGMM algorithm to be ineffective due the above rationale, we only evaluated FoolsGold and Krum for the rest of the experiments for this attack.

Through our i.i.d. experiments, we demonstrate that with a differing distribution of the data set across endpoints, the attack remains effective. We observe that the results for this case are less optimal than the non-i.i.d. case. However, this is primarily a result of our reuse of toggle parameters (β, γ) from the grid search on the non-i.i.d. data. We observe for that for the cases of 50% and 80% Adv., the attack remains very successful by achieving mean ASRs above 0.75. On the other hand, the 30% Adv. fails as a result of the less optimal choice of toggle parameters, showing that Sybils can compensate for bad choices in toggle parameters. Moreover, we find that the Multi-Krum algorithm achieves better performance in Byzantine cases within the i.i.d. environment. This is due to the relative distributions of data, where gradients produced by honest nodes can be vastly similar to each other, which differs from the gradients of the adversaries due to use of poisoned data. Since Krum observes the distance between gradients, this factor makes honest and adversarial gradients easily separable when the Byzantine condition holds within an i.i.d. environment.

We demonstrate that the proposed attack remains effective within the more complex environment of learning the CIFAR-10 object detection using a convolutional neural network. The same general trends as in the non-i.i.d. MNIST experiments are observed, however, the cases for a single adversary prove less effective. The other notable observation is the lower ASR for FoolsGold with 80% adversaries when compared with 50% adversaries. This is due to the adversaries having to toggle off for longer periods of time, as with a larger model, FoolsGold is able to better detect adversary's gradient during their collusion from being toggled on. This effect is observed here as the larger gradients allow for a greater dimensionality for the gradients to be similar. However, the curse of dimensionality makes the more subtle (smaller) attacks harder to detect while large scale attacks becoming more detectable. Smaller or the subtlety refers to the number of adversaries performing it.

With our experiments with the KDD Cup '99 data set, we found that the balance of true class data points impacts the effectiveness of the attack. For example, in a network com-

Data set	Algorithm	Mean ASR (STD ASR)		
		10% Adv.	30% Adv.	50% Adv.
MNIST	FoolsGold	0.998 (0.0446)	0.351 (0.477)	0.402 (0.49)
	Krum	0.493 (0.5)	0.383 (0.4)	0.365 (0.39)
	STD-DAGMM	0.481 (0.5)	0.641 (0.442)	0.72 (0.409)
CIFAR-10	FoolsGold	0.998 (0.0446)	0.431 (0.495)	0.307 (0.461)
	Krum	0.495 (0.5)	0.275 (0.351)	0.379 (0.406)
	STD-DAGMM	0.459 (0.498)	0.665 (0.438)	0.702 (0.405)
KDD Cup '99	FoolsGold	0 (0)	0 (0)	0 (0)
	Krum	0.335 (0.454)	0.35 (0.364)	0.453 (0.459)
	STD-DAGMM	0.208 (0.404)	0.464 (0.42)	0.6 (0.45)

Table 9: Results of the On-Off Free Riding Experiments

posed of 50% on-off label flipping adversaries which flip the "smurf" class to the "normal" class, only an ASR of 0.000916 (0.000268) is achieved. For this case there are more samples from the "smurf" class than the "normal" class, however, we achieve the greater results presented in Table 4.2 by flipping the "back" class into the "normal" class, since "back" has less samples than "normal". Furthermore, we show that this property is not merely the result of the chosen class flip through our experiments on non-i.i.d. where we flip the class 8 samples to 2 rather than 0 to 1 and achieve an effective attack. As with the non-i.i.d. experiments, our results are less optimal for this case as we reuse the toggle parameters found through a grid search for the 0 to 1 on-off label flip attack. The property of Sybils compensating for a bad selection of toggle parameter emerges in this case as well.

We have compared the effectiveness of our attack with the canonical label flipping attack and scaled backdoor attack (Bagdasaryan et al., 2020). Our results show that on-off attack is substantially more effective than either of those attacks, simply due to defeating history aggregating mechanisms. Furthermore, we observe that the Krum algorithm remains ineffective within the non-i.i.d. environments. Additionally, the results from the STD-DAGMM experiment with scaled backdoor attacks are omitted, as the GMM mechanism, being unable to detect sufficient variance within the gradients, completely fails to detect the attack.

Appendix C.2 On-off Free riding

For this environment, the on-off free riding attack exhibits many of the same properties as the on-off label flipping attack. These shared properties are induced by the on-off mechanism to evade the history aggregation mechanisms of FoolsGold and STD-DAGMM. Ultimately, we show in Table Appendix C.2, that the on-off variant of free rider attacks is mostly successful against FoolsGold, and is successful in all cases against the Multi-Krum and STD-DAGMM algorithms. However, the former two algorithms present some mitigation of the attacks. This is observable through ASR values with a mean below 0.5 and standard

deviation close to 0.5, which show that that the adversaries only free ride for less than half of the system runtime. Conversely, the STD-DAGMM results are particularly surprising due to having been designed to mitigate free-riding attacks. That is, as shown through mean ASR values above 0.5, the on-off toggling fools the STD-DAGMM learning mechanism, thereby rendering the defense mechanism ineffective.

FoolsGold is generally unable to completely mitigate the attack, due to the on-off attack exploiting the history mechanisms of the algorithm. However, we find an exception to the effectiveness of the attack against FoolsGold. This occurs with the unbalanced class representation in KDD Cup'99. In this case, the algorithm detects all the free riders despite on-off toggling. This emerges from the properties of the gradients generated in these cases. In this case, the adversaries' gradients are distinctly unique from any endpoint's and are therefore easily detected by the FoolsGold algorithm.

The attack generally has a moderate effectiveness against Multi-Krum, where the Byzantine collusion cases are more effectively mitigated. These patterns emerge through the functionality of Krum not being suited to the mitigation of free-rider attacks, as Krum merely selects gradients least distant from each of the other. Since the global update from the previous round will typically remain close to that of the current round's individual endpoint updates, the free-rider's will consistently evade the Krum algorithm. However, Krum remains relatively successful in detecting the collusion of Byzantine on-off free-riders; nonetheless it never achieves complete mitigation of the attack.

Despite being designed for the detection of free-rider attacks, we find the STD-DAGMM algorithm is most adversely effected by the on-off attack. Our results show that the impact of the on-off toggling extends into systems that learn from historical gradients, that is, in addition to those that aggregate.

Appendix C.3 Mouthing

In Table Appendix C.3, we present the results from our experiments with good mouthing and bad mouthing adversaries.

The FoolsGold algorithm effectively detects the adversaries and mitigates them. However, in most cases this is to the detriment of the victim, as they are grouped together as Sybils with the adversaries. This flaw can be remedied by the application of a policy, which we will discuss in Section 5. Moreover, the imbalanced class proportions of the KDD Cup '99 data set again induces unique properties for FL attacks. In this case, the algorithm effectively singles out the adversaries from the victim, and thus correctly mitigates the attack. The class imbalance further highlights the adversaries, as their updates being copies of the victim's are balanced in comparison to each other. The other factor that enables the detection of the adversaries is the noise that is being applied to their updates. If present, this makes the adversaries' updates sufficiently different from victim but not enough from each other, since the noise is retrieved from the same distribution. FoolsGold performs similar effective mitigation for the bad mouthing attack. However, in this case, the algorithm specifically detects adversaries through its ability to distinguish between their updates and victims, as adversary updates are the negation of the victims.

The Multi-Krum algorithm was adversely effected by these attacks to a greater extent with respect to the number of adversaries. As Krum prioritizes gradients that are close to each

Environment	Algorithm	Mean ASR (STD ASR)		
		10% Adv.	30% Adv.	50% Adv.
MNIST (GM)	FoolsGold	$p = 0$	$p = 0$	$p = 0$
	Krum	0.584 (0.269)	0.656 (0.183)	0.795 (0.0743)
	STD-DAGMM	0.829 (0.135)	0.766 (0.183)	0.812 (0.12)
MNIST (BM)	FoolsGold	$p = 0$	$p_{\mathcal{S}} = 0$	$p_{\mathcal{S}} = 0$
	Krum	$p \gg 1$	$p \gg 1$	$p \gg 1$
	STD-DAGMM	$p \gg 1$	$p \gg 1$	$p \gg 1$
CIFAR-10	FoolsGold	$p = 0$	$p = 0$	$p = 0$
	Krum	0.903 (0.113)	0.895 (0.125)	0.82 (0.196)
	STD-DAGMM	0.838 (0.135)	0.88 (0.134)	0.851 (0.157)
KDD Cup '99	FoolsGold	$p = 0$	$p_{\mathcal{S}} = 0$	$p_{\mathcal{S}} = 0$
	Krum	0.922 (0.0843)	0.91 (0.0726)	0.947 (0.0383)
	STD-DAGMM	0.93 (0.0955)	0.967 (0.0226)	0.976 (0.0178)

Table 10: Results of the Good and Bad Mouthing Experiments

other, the adversaries are given greater influence over the system, as the attack's basis is on copying the victim's gradient. Our results for the bad mouthing attack on Krum show a complete success which arises from the overflow flaw in the algorithm.

As in the case of the label flipping attack, STD-DAGMM is unable to detect or effectively mitigate the mouthing attacks. That is, due to the attack directly using the victim's gradient, which is legitimate. Therefore STD-DAGMM allows adversarial gradients since they are copies of a legitimate gradient, and the algorithm is only designed for the detection of illegitimate gradients that are not produced through training. Since the algorithm is unable to detect the attacks, the same parameter overflow issue from the victim gradients as with the Multi-Krum algorithm occurs. This is reflected in the results with $p \gg 1$ due to the complete success of attack.

Appendix C.4 Viceroy

In Table Appendix C.4, we have show that the proposed Viceroy algorithm remains effective in mitigating the on-off label flipping, on-off free rider, label flipping as well as good mouthing and bad mouthing attacks for the 10 endpoint environment.

The Viceroy algorithm remains susceptible to constant attacks involving a single adversary submitting a unique gradient, as seen for example, in the 10% Adv. label flipping. This is a flaw that persists from the underlying FoolsGold algorithm. However, Fung et al. (2020) present a solution using an algorithm designed for the detection of single/Byzantine adversaries such as Multi-Krum. This solution is also applicable to our algorithm, by adding into Line 3 of Algorithm 1 a non-sybil FLDS that additionally scales the value of p.

Attack	Data set	Mean ASR (STD ASR)		
		10% Adv.	30% Adv.	50% Adv.
On-off Label Flip	MNIST	6.11e-5 (3.03e-4)	9.57e-5 (3.49e-4)	2e-6 (4.6e-5)
	CIFAR-10	0.0803 (0.0545)	0.106 (0.0808)	0.122 (0.0925)
	KDD Cup '99	0.978 (0.147)	0.984 (0.125)	0.982 (0.133)
On-off Free Ride	MNIST	0 (0)	5.99e-3 (0.0772)	0 (0)
	CIFAR-10	0.012 (0.109)	2e-3 (0.0446)	2e-3 (0.0446)
	KDD Cup '99	0 (0)	0 (0)	0 (0)
Good Mouthing	MNIST	$p = 0$	$p = 0$	$p = 0$
	CIFAR-10	$p = 0$	$p = 0$	$p = 0$
	KDD Cup '99	$p = 0$	$p = 0$	$p = 0$
Bad Mouthing	MNIST	0.814 (0.595)	$p = 0$	$p = 0$
	CIFAR-10	0.928 (0.0823)	$p_{\mathcal{S}} = 0$	$p_{\mathcal{S}} = 0$
	KDD Cup '99	$p = 0$	$p = 0$	$p = 0$
Label Flipping	MNIST	0.496 (0.317)	0 (0)	4e-6 (6.4e-5)
	CIFAR-10	0 (0)	0 (0)	0 (0)
	KDD Cup '99	0.98 (0.14)	0.996 (0.0631)	0.996 (0.0631)
Scaled Backdoor	MNIST	0.514 (0.303)	0 (0)	1e-5 (1e-4)
	CIFAR-10	0 (0)	0 (0)	0 (0)
	KDD Cup '99	0.981 (0.135)	0.993 (0.0809)	0.995 (0.0656)

Table 11: Results of the Viceroy Experiments

The algorithm otherwise provides similar properties to those observed within the larger environment.

Appendix Appendix D. Computing the Viceroy Parameters

In this section, we provide equations for the calculation of the parameters used with the reputation update mechanism of Viceroy. They are each defined to take as input, the number of rounds required for their specified functionality.

The first parameter is ω which defines the rate of decay of the reputation value towards the neutral value of 0. It is calculated using (17) where τ_0 is the chosen number of rounds to reach zero, and ϵ is a parameter value close to zero that defines the granularity of the system.[6] This equation assumes a worst-case scenario where endpoint contributions do not assist in increasing the reputation.

$$\omega = \sqrt[\tau_0]{|\epsilon|} \tag{17}$$

6. For our implementation, we use python's `sys.float_info.epsilon`

The other calculated parameter is η which defines the rate of increase of reputation under a best case scenario where the aggregated reputation experiences minimal decay. It is calculated using (18) where τ_1 is the chosen number of rounds for the reputation head from the neural value of 0 to the maximal value of 1.

$$\eta = \frac{2}{\tau_1} \tag{18}$$

Appendix D.1 On our Selections of the Parameters

For our experiments with the Viceroy algorithm, we select $\tau_0 = 56$ and $\tau_1 = 10$. We can observe from the proofs given in Appendix B that that these parameters provide reasonably pessimistic assumptions about the effectiveness of poisoning upon the federated learning process. In the following, we briefly discuss the minimum bounds created by the selected parameters. We assume that the current model is a single unit of distance from the adversarial target model, $\|\theta^{(0)} - \theta^a\| = 1$, and that the distance is approximately zero after the model has been poisoned for $\mathcal{T}$ rounds. Note that the learning rate, n, is set to 0.1 for our experiments.

By observing the bounds provided by (16) we see that the value of $\tau_0 = 56$ under (17) yields $\mathcal{T} = 0.1787$ stating that only a single round is sufficient for unmitigated poisoning to occur. While observing the bounds provided by (15) we see that the value of $\tau_1 = 10$ under (18) yields $\epsilon = 1$ stating that an adversarial objective value of 1 is sufficient to perform an effective unmitigated poisoning attack.

References

Ibrahim SI Abuhaiba and Huda B Hubboub. Reinforcement swap attack against directed diffusion in wireless sensor networks. *International Journal of Computer Network and Information Security*, 5(3):13–24, 2013. `http://www.mecs-press.org/ijcnis/ijcnis-v5-n3/IJCNIS-V5-N3-2.pdf`.

Manoj Ghuhan Arivazhagan, Vinay Aggarwal, Aaditya Kumar Singh, and Sunav Choudhary. Federated learning with personalization layers. *ArXiv*, abs/1912.00818, 2019. `https://arxiv.org/abs/1912.00818`.

Giuseppe Ateniese, Luigi V Mancini, Angelo Spognardi, Antonio Villani, Domenico Vitali, and Giovanni Felici. Hacking smart machines with smarter ones: How to extract meaningful data from machine learning classifiers. *International Journal of Security and Networks*, 10(3):137–150, 2015. `https://arxiv.org/abs/1306.4447`.

Sana Awan, Bo Luo, and Fengjun Li. CONTRA: Defending against poisoning attacks in federated learning. In *European Symposium on Research in Computer Security*, pages 455–475. Springer, 2021. `https://www.ittc.ku.edu/~bluo/pubs/Awan2021ESORICS.pdf`.

Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How to backdoor federated learning. In *International Conference on Artificial Intelligence and Statistics*, pages 2938–2948. PMLR, 2020. `https://arxiv.org/abs/1807.00459`.

Arjun Nitin Bhagoji, Supriyo Chakraborty, Prateek Mittal, and Seraphin Calo. Analyzing federated learning through an adversarial lens. In *International Conference on Machine Learning*, pages 634–643. PMLR, 2019. https://arxiv.org/abs/1811.12470.

Battista Biggio, Blaine Nelson, and Pavel Laskov. Poisoning attacks against support vector machines. In *International Conference on Machine Learning*, 2012. https://arxiv.org/abs/1206.6389.

Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. Machine learning with adversaries: Byzantine tolerant gradient descent. *Advances in Neural Information Processing Systems*, 30, 2017. https://papers.nips.cc/paper/2017/hash/f4b9ec30ad9f68f89b29639786cb62ef-Abstract.html.

Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In *ACM SIGSAC Conference on Computer and Communications Security*, pages 1175–1191, 2017. https://dl.acm.org/doi/10.1145/3133956.3133982.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. http://github.com/google/jax.

Wei Chen, Kartikeya Bhardwaj, and Radu Marculescu. FedMAX: mitigating activation divergence for accurate and communication-efficient federated learning. *ArXiv*, 2020. https://arxiv.org/abs/2004.03657.

John R. Douceur. The sybil attack. In *International workshop on peer-to-peer systems*, pages 251–260. Springer, 2002. https://www.microsoft.com/en-us/research/publication/the-sybil-attack/.

Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. http://archive.ics.uci.edu/ml.

Úlfar Erlingsson, A. Korolova, and V. Pihur. RAPPOR: Randomized aggregatable privacy-preserving ordinal response. In *ACM SIGSAC conference on computer and communications security*, pages 1054–1067, 2014. https://arxiv.org/abs/1407.6981.

Matthew Fredrikson, Eric Lantz, Somesh Jha, Simon Lin, David Page, and Thomas Ristenpart. Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing. In *Proceedings of the 23rd USENIX Security Symposium (USENIX Security 14)*, pages 17–32, 2014. https://dl.acm.org/doi/10.5555/2671225.2671227.

Matthew Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, pages 1322–1333, 2015. https://dl.acm.org/doi/10.1145/2810103.2813677.

Clement Fung, Jamie Koerner, Stewart Grant, and Ivan Beschastnikh. Dancing in the dark: Private multi-party machine learning in an untrusted setting. *ArXiv*, 2018. https://arxiv.org/abs/1811.09712.

Clement Fung, Chris J. M. Yoon, and Ivan Beschastnikh. The limitations of federated learning in sybil settings. In *Symposium on Research in Attacks, Intrusion, and Defenses*, RAID, 2020. https://www.usenix.org/conference/raid2020/presentation/fung.

Robin C. Geyer, T. Klein, and Moin Nabi. Differentially private federated learning: A client level perspective. *ArXiv*, abs/1712.07557, 2017. https://arxiv.org/abs/1712.07557.

Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. BadNets: Identifying vulnerabilities in the machine learning model supply chain. *ArXiv*, abs/1708.06733, 2017. https://arxiv.org/abs/1708.06733.

Stephen Hardy, Wilko Henecka, Hamish Ivey-Law, Richard Nock, Giorgio Patrini, Guillaume Smith, and Brian Thorne. Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption. *ArXiv*, abs/1711.10677, 2017. https://arxiv.org/abs/1711.10677.

Lie He, Sai Praneeth Reddy Karimireddy, and Martin Jaggi. Byzantine-robust learning on heterogeneous datasets via resampling. *ArXiv*, abs/2006.09365, 2020. https://arxiv.org/abs/2006.09365.

Yingzhe He, Guozhu Meng, Kai Chen, Xingbo Hu, and Jinwen He. Towards privacy and security of deep learning systems: A survey. *ArXiv*, abs/1911.12562, 2019. https://arxiv.org/abs/1911.12562.

Briland Hitaj, Giuseppe Ateniese, and Fernando Pérez-Cruz. Deep models under the GAN: Information leakage from collaborative deep learning. *ACM SIGSAC Conference on Computer and Communications Security*, pages 603–618, 2017. https://arxiv.org/abs/1702.07464.

Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. Measuring the effects of non-identical data distribution for federated visual classification. *ArXiv*, 2019. https://arxiv.org/abs/1909.06335.

Bargav Jayaraman, Lingxiao Wang, David Evans, and Quanquan Gu. Revisiting membership inference under realistic assumptions. *ArXiv*, abs/2005.10881, 2020. https://arxiv.org/abs/2005.10881.

Nikhil U. Joshi and Rewanth Tammana. GDALR: An efficient model duplication attack on black box machine learning models. In *IEEE International Conference on System, Computation, Automation and Networking*, pages 1–6. IEEE, 2019. https://ieeexplore.ieee.org/document/8878726.

Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. SCAFFOLD: Stochastic controlled averaging for federated learning. In *International Conference on Machine Learning*, pages 5132–5143. PMLR, 2020. https://proceedings.mlr.press/v119/karimireddy20a.html.

Auguste Kerckhoffs. La cryptographie militaire. *Journal des sciences militaires*, IX:5–38, 1883a.

Auguste Kerckhoffs. La cryptographie militaire. *Journal des sciences militaires*, IX:161–191, 1883b.

Jakub Konečný, H. Brendan McMahan, and Daniel Ramage. Federated optimization: Distributed optimization beyond the datacenter. *ArXiv*, abs/1511.03575, 2015. https://arxiv.org/abs/1511.03575.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. *Master's thesis, Department of Computer Science, University of Toronto*, 2009. https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.

Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *Journal of the ACM*, 30(3):668–676, 1983. https://www.microsoft.com/en-us/research/publication/byzantine-generals-problem/.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998a. https://ieeexplore.ieee.org/document/726791.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998b. https://ieeexplore.ieee.org/document/726791.

Jeffrey Li, Mikhail Khodak, Sebastian Caldas, and Ameet Talwalkar. Differentially private meta-learning. *ArXiv*, abs/1909.05830, 2020. https://arxiv.org/abs/1909.05830.

Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *ArXiv*, 2018. https://arxiv.org/abs/1812.06127.

Jierui Lin, Min Du, and Jian Liu. Free-riders in federated learning: Attacks and defenses. *ArXiv*, abs/1911.12560, 2019. https://arxiv.org/abs/1911.12560.

Xiang Mao and Janise McNair. Effect of on/off misbehavior on overhearing based cooperation scheme for MANET. *Military Communications Conference*, pages 1086–1091, 2010. https://ieeexplore.ieee.org/document/5680088.

Paul Mason. The racist hijacking of microsoft's chatbot shows how the internet teems with hate. *The Guardian*, 2016. https://www.theguardian.com/world/2016/mar/29/microsoft-tay-tweets-antisemitic-racism.

Luis Muñoz-González, Kenneth T Co, and Emil C Lupu. Byzantine-robust federated machine learning through adaptive model averaging. *ArXiv*, 2019. https://arxiv.org/abs/1909.05125.

Thien Duc Nguyen, Phillip Rieger, Huili Chen, Hossein Yalame, Helen Möllering, Hossein Fereidooni, Samuel Marchal, Markus Miettinen, Azalia Mirhoseini, Shaza Zeitouni, Farinaz Koushanfar, Ahmad-Reza Sadeghi, and Thomas Schneider. FLAME: Taming backdoors in federated learning. In *Proceedings of the 31st USENIX Security Symposium (USENIX Security 22)*, pages 1415–1432, Boston, MA, August 2022. USENIX Association. ISBN 978-1-939133-31-1. URL https://www.usenix.org/conference/usenixsecurity22/presentation/nguyen. https://arxiv.org/abs/2101.02281.

Nicolas Papernot, P. McDaniel, Arunesh Sinha, and Michael P. Wellman. Towards the science of security and privacy in machine learning. *ArXiv*, abs/1611.03814, 2016. https://arxiv.org/abs/1611.03814.

Robert Nikolai Reith, Thomas Schneider, and Oleksandr Tkachenko. Efficiently stealing your machine learning models. *Proceedings of the 18th ACM Workshop on Privacy in the Electronic Society*, pages 198–210, 2019. https://dl.acm.org/doi/10.1145/3338498.3358646.

Ahmed Salem, Yibao Zhang, Mathias Humbert, Mario Fritz, and Michael Backes. ML-Leaks: Model and data independent membership inference attacks and defenses on machine learning models. *ArXiv*, abs/1806.01246, 2019. https://arxiv.org/abs/1806.01246.

Claude E Shannon. Communication theory of secrecy systems. *The Bell system technical journal*, 28(4):656–715, 1949. https://ieeexplore.ieee.org/document/6769090.

Shiqi Shen, Shruti Tople, and Prateek Saxena. Auror: Defending against poisoning attacks in collaborative deep learning systems. In *Proceedings of the 32nd Annual Conference on Computer Security Applications*, pages 508–519, 2016. https://dl.acm.org/doi/10.1145/2991079.2991125.

Reza Shokri and Vitaly Shmatikov. Privacy-preserving deep learning. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, pages 1310–1321, 2015. https://ieeexplore.ieee.org/document/7447103.

Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. *IEEE Symposium on Security and Privacy*, pages 3–18, 2017. https://arxiv.org/abs/1610.05820.

Florian Tramèr, Fan Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. Stealing machine learning models via prediction APIs. In *Proceedings of the 25th USENIX Security Symposium (USENIX Security 16)*, pages 601–618, 2016. https://arxiv.org/abs/1609.02943.

Jane WakeField. Microsoft chatbot is taught to swear on twitter. *BBC News*, 2016. https://www.bbc.com/news/technology-35890188.

Wei-Chung Wan, Jianrong Lu, Shengshan Hu, Leo Yu Zhang, and Xiaobing Pei. Shielding federated learning: A new attack approach and its defense. *IEEE Wireless Communications and Networking Conference*, pages 1–7, 2021. https://ieeexplore.ieee.org/document/9417334.

Hongyi Wang, Kartik Sreenivasan, Shashank Rajput, Harit Vishwakarma, Saurabh Agarwal, Jy-yong Sohn, Kangwook Lee, and Dimitris Papailiopoulos. Attack of the tails: Yes, you really can backdoor federated learning. *ArXiv*, 2020. https://arxiv.org/abs/2007.05084.

Ning Wang, Xiaokui Xiao, Yin Yang, Jun Zhao, Siu Cheung Hui, Hyejin Shin, Junbum Shin, and Ge Yu. Collecting and analyzing multidimensional data with local differential privacy. In *IEEE 35th International Conference on Data Engineering*, pages 638–649. IEEE, 2019. https://arxiv.org/abs/1907.00782.

Kang Wei, Jun Li, Ming Ding, Chuan Ma, Howard H Yang, Farhad Farokhi, Shi Jin, Tony QS Quek, and H Vincent Poor. Federated learning with differential privacy: Algorithms and performance analysis. *IEEE Transactions on Information Forensics and Security*, 15:3454–3469, 2020. https://ieeexplore.ieee.org/document/9069945.

Qi Xia, Zeyi Tao, Zijiang Hao, and Qun Li. FABA: An algorithm for fast aggregation against byzantine attacks in distributed neural networks. *International Joint Conferences on Artificial Intelligence*, 2019. doi: 10.24963/ijcai.2019/670. https://par.nsf.gov/biblio/10119268.

Chulin Xie, Keli Huang, Pin-Yu Chen, and Bo Li. DBA: Distributed backdoor attacks against federated learning. In *International Conference on Learning Representations*, 2019. https://openreview.net/forum?id=rkgyS0VFvr.

Cong Xie, Oluwasanmi Koyejo, and Indranil Gupta. Phocas: Dimensional byzantine-resilient stochastic gradient descent. *Journal of Environmental Sciences (China) English Ed*, 2018. https://arxiv.org/abs/1805.09682.

Dong Yin, Yudong Chen, Ramchandran Kannan, and Peter Bartlett. Byzantine-robust distributed learning: Towards optimal statistical rates. In *International Conference on Machine Learning*, pages 5650–5659. PMLR, 2018. https://proceedings.mlr.press/v80/yin18a.html.

Chengliang Zhang, Suyi Li, Junzhe Xia, Wei Wang, Feng Yan, and Yang Liu. BatchCrypt: Efficient homomorphic encryption for cross-silo federated learning. In *USENIX Annual Technical Conference (USENIXATC 20)*, pages 493–506, 2020. https://www.usenix.org/conference/atc20/presentation/zhang-chengliang.