Securing Federated Sensitive Topic Classification against Poisoning Attacks

Tianyue Chu

IMDEA Networks Institute,
Universidad Carlos III de Madrid

Alvaro Garcia-Recuero

IMDEA Networks Institute

Costas Iordanou

Cyprus University of Technology

Georgios Smaragdakis

TU Delft

Nikolaos Laoutaris

IMDEA Networks Institute

Abstract—**We present a Federated Learning (FL) based solution for building a distributed classifier capable of detecting URLs containing sensitive content, i.e., content related to categories such as health, political beliefs, sexual orientation, etc. Although such a classifier addresses the limitations of previous offline/centralised classifiers, it is still vulnerable to poisoning attacks from malicious users that may attempt to reduce the accuracy for benign users by disseminating faulty model updates. To guard against this, we develop a robust aggregation scheme based on subjective logic and residual-based attack detection. Employing a combination of theoretical analysis, trace-driven simulation, as well as experimental validation with a prototype and real users, we show that our classifier can detect sensitive content with high accuracy, learn new labels fast, and remain robust in view of poisoning attacks from malicious users, as well as imperfect input from non-malicious ones.**

I. INTRODUCTION

Most people are not aware that tracking services are present even on sensitive web domains. Being tracked on a cancer discussion forum, a dating site, or a news site with non-mainstream political affinity can be considered an "elephant in the room" when it comes to the anxieties that many people have about their online privacy. The General Data Protection Regulation (GDPR) [32] puts specific restrictions on the collection and processing of sensitive personal data *"revealing racial or ethnic origin, political opinions, religious or philosophical beliefs, or trade union membership, also genetic data, biometric data for the purpose of uniquely identifying a natural person, data concerning health or data concerning a natural persons sex life or sexual orientation"*. So do other public bodies around the world, *e.g.* in California (California Consumer Privacy Act (CCPA) [33]), Canada [34], Israel [35], Japan [36], and Australia [37].

In a recent paper, Matic et al. [4] showed how to train a classifier for detecting whether the content of a URL relates to any of the above-mentioned sensitive categories. The classifier was trained using 156 thousand sensitive URLs obtained from the Curlie [31] crowdsourced web taxonomy project. Despite the demonstrated high accuracy, this method has limitations that stem from being *centralised* and *tied to a fixed training set*. The first limitation means that the method cannot be used "as is" to drive a privacy-preserving distributed classification system. The second limitation implies that it is not straightforward to cover new labels related to yet unseen sensitive content. For example, in their work the Health category could be classified with accuracy greater than 90%. However, the training labels obtained from Curlie in 2020 did not include any labels related to the COVID-19 pandemic. Therefore, as will be shown later, this classifier classifies COVID-19 related sites with only 53.13% accuracy.

Federated Learning (FL) [5], [12] offers a natural solution to the above two mentioned limitations, namely, centralized training and training for a fixed training set. FL allows different clients to train their classification models locally without revealing new or existing sensitive URLs that they label, while collaborating by sharing model updates that can be combined to build a superior global classification model. FL has proved its value in a slew of real-world applications, ranging from mobile computing [45]–[47] to health and medical applications [48]–[50]. However, due to its very nature, FL is vulnerable to so-called *poisoning attacks* [11], [25] mounted by malicious clients that may intentionally train their local models with faulty labels or backdoor patterns, and then disseminate the resulting updates with the intention of reducing the classification accuracy for other benign clients. State-of-the-art approaches for defending against such attacks depend on robust aggregation [14], [15], [19], [26], [59] which, as we will demonstrate later, are slow to converge, thereby making them impractical for the sensitive-content classification problem that we tackle in this paper.

Our Contributions: In this paper, we employ FL for sensitive content classification. We show how to develop a robust FL method for classifying arbitrary URLs that may contain GDPR sensitive content. Such a FL-based solution allows building a distributed classifier that can be offered to end-users in the form of a web browser extension in order to: (i) warn them before and while they navigate into such websites, especially when they are populated with trackers, and (ii) allow them to contribute new labels, e.g., health-related websites about COVID-19, and thus keeping the classifier always up-to-date. To the best of our knowledge this method represents the first use of FL for such task.

Our second major contribution is the development of a reputation score for protecting our FL-based solution from poisoning attacks [11], [25]. Our approach is based on a novel combination of subjective logic [3] with residual-based attack detection. Our third contribution is the development of an extensive theoretical and experimental performance evaluation framework for studying the accuracy, convergence, and resilience to attacks of our proposed mechanism. Our final contribution is the implementation of our methods in a prototype system called *EITR* (standing for "Elephant In the Room" of privacy) and our preliminary experimental validation with real users tasked to provide fresh labels for the accurate classification of COVID-19 related URLs.

Our findings: Using a combination of theoretical analysis, simulation, and experimentation with real users, we:

• Demonstrate experimentally that our FL-based classifier achieves comparable accuracy with the centralised one presented in [4].
• Prove analytically that under data poisoning attacks, our reputation-based robust aggregation built around subjective logic, converges to a near-optimal solution of the corresponding Byzantine fault tolerance problem under standard assumptions. The resulting performance gap is determined by the percentage of malicious users.
• Evaluate experimentally our solution against state-of-the-art algorithms such as Federated Averaging [5], Coordinate-wise median [19], Trimmed-mean [19], FoolsGold [14], Residual-based re-weighting [15] and FLTrust [59], and show that our algorithm is robust under Byzantine attacks by using different real-world datasets. We demonstrate that our solution outperforms these popular solutions in terms of convergence speed by a factor ranging from $1.6\times$ to $2.4\times$ while achieving the same or better accuracy. Furthermore, our method yields the most consistent and lowest Attack Success Rate (ASR), with at least 72.3% average improvement against all other methods.
• Validate using our *EITR* browser extension that our FL-based solution can quickly learn to classify health-related sites about COVID-19, even in view of noisy/inconsistent input provided by real users.

The remainder of the article is structured as follows: Section II introduces the background for our topic. Section III presents our reputations scheme for FL-based sensitive content classification, as well as its theoretical analysis and guarantees. Section IV covers our extensive performance evaluation against the state-of-the-art and Section V some preliminary results from our *EITR* browser extension. Section VI concludes the paper and points to on-going and future work including the generalization of our method to other topics.

II. BACKGROUND

A. A Centralised Offline Classifier for Sensitive Content

Matic et al. [4] have shown how to develop a text classifier able to detect URLs that contain sensitive content. This classifier is centralised and was developed in order to conduct a one-off offline study aimed at estimating the percentage of the web that includes such content. Despite achieving an accuracy of at least 88%, utilising a high-quality training set meticulously collected by filtering the Curlie web-taxonomy

project [31], this classifier cannot be used "as is" to protect real users visiting sensitive URLs populated by tracking services.

B. Challenges in Developing a Practical Classifier for Users

From offline to online: The classifier in [4] was trained using a dataset of 156 thousand sensitive URLs. Despite being the largest dataset of its type in recent literature, this dataset is static and thus represents sensitive topics up to the time of its collection. This does not mean, of course, that a new classifier trained with this data would never be able to accurately classify new URLs pertaining to those sensitive categories. This owes to the fact that categories such as Health, involve content and terms that do not change radically with time. Of course, new types of sensitive content may appear that, for whatever reason, may not be so accurately classified using features extracted from past content of the same sensitive category. Content pertaining to the recent COVID-19 pandemic is such an example. Although the Health category had 74,764 URLs in the training set of [4] which lead to a classification accuracy of 88% for Health, as we will see later in Figure 14 middle of Section V-C, the classifier of [4] classifies accurately as Health only 53.13% of the COVID-19 URLs with which we tested it. This should not come as a surprise since the dataset of [4] corresponds to content generated before the first months of 2020, during which COVID-19 was not yet a popular topic. Therefore, we need to find a way to update an existing classifier so that it remains accurate as new sensitive content appears.

From centralised to distributed: A natural way to keep a classifier up-to-date is to ask end-users to label new sensitive URLs as they encounter them. End-users can report back to a centralised server such URLs which can then be used to retrain the classification model. This, however, entails obvious privacy challenges of "Catch-22" nature, since to protect users by warning them about the presence of trackers on sensitive URLs, they would first be required to report to a potentially untrusted centralised server that they visit such URLs. Even by employing some methods for data scarcity, e.g., semi-supervised learning, the manual labelling from users remains sensitive and may be harmed by the untrusted server. Federated Learning, as already mentioned, is a promising solution for avoiding the above Catch22 by conducting a distributed, albeit, privacy-preserving, model training. In an FL approach to our problem, users would label new URLs locally, e.g., a COVID-19 URL as Health, retrain the classifier model locally, and then send model updates, not labelled data, to a centralised server that collects such updates from all users, compiles and redistributes the new version of the model back to them. In Section III we show how to develop a distributed version of the sensitive topic classifier of [4] using FL. The trade-off of using FL, is that the distributed learning group becomes vulnerable to attacks, such as "label-flipping" poisoning attacks discussed in Section IV. This paper develops a reputation scheme for mitigating such attacks. Other types of attacks and measures for preserving the privacy of users that participate in a FL-based classification system for sensitive content are discussed in Section VI.

C. Related Work

Privacy preserving crowdsourcing: Similar challenges to the ones discussed in the previous paragraph have been faced in

services like the *Price $heriff* [53] and *eyeWnder* [54] that have used crowdsourcing to detect online price discrimination and targeted advertising, respectively. Secure Multi-Party Computation (SMPC) techniques such as private k-means [55] are used to allow end-users to send data in a centralised server in a privacy-preserving manner. The centralised computation performed by *Price $heriff* and *eyeWnder* is not of ML nature, thus leaving data anonymisation as the main challenge, for which SMPC is a good fit. Classifying content as sensitive or not is a more complex ML-based algorithm for which FL is a more natural solution than SMPC.

General works on FL: FL [5], [12] is a compelling technique for training large-scale distributed machine learning models while maintaining security and privacy. The motivation for FL is that local training data is always kept by the clients and the server has no access to the data. Due to this benefit that alleviates privacy concerns, several corporations have utilised FL in real world services. In mobile devices, FL is used to predict keyboard input [45], human mobility [46] and behaviour for the Internet of Things [47]. FL is also applied in healthcare to predict diseases [48], [49], detect patient similarity [50] while overcoming any privacy constrains. For the classification, FL is not only implemented for image classification [51] but also text classification [52].

Resilience to poisoning attacks: Owing to its nature [11], [25], FL is vulnerable to poisoning attacks, such as label flipping [15] and backdoor attacks [11]. Therefore, several defence methods have been developed [14], [15], [19]. While these state-of-the-art approaches perform excellently in some scenarios, they are not without limitations. First, they are unsuitable for our sensitive content classification, which necessitates that a classifier responds very fast to "fresh" sensitive information appearing on the Internet. In existing methods, the primary objective is to achieve a high classification accuracy. This is achieved via statistical analysis of client-supplied model updates and discarding of questionable outliers before the aggregation stage. However, since the server distrusts everyone by default, even if an honest client discovers some fresh sensitive labels, its corresponding updates may be discarded or assigned low weights, up until more clients start discovering these labels. This leads to a slower learning rate for new labels.

Second, recent studies [11], [25] have shown that existing Byzantine-robust FL methods are still vulnerable to local model poisoning since they are forgetful by not tracking information from previous aggregation rounds. Thus, an attacker can efficiently mount an attack by spreading it across time [30]. For example, [21] recently showed that even after infinite training epochs, any aggregation which is neglectful of the past cannot converge to an efficient solution.

The preceding studies demonstrate the importance of incorporating clients' previous long-term performance in evaluating their trustworthiness. Few recent studies have considered this approach [21], [59]. In [21], the authors propose leveraging historical information for optimisation, but not for assessing trustworthiness. In [59], a trust score is assigned to each client model update according to the cosine similarity between the client's and server's model updates, which is trained on the server's root dataset (details in Section IV-A3). However, it is impractical for a server to obtain additional data, such as a root dataset, in order to train a server-side model. In addition, because the server collects root data only once and

TABLE I: Notation

Abbreviation	Description
M	the total number of clients
N	the number of parameters of global model
Q	the number of samples of each client
T	the total number of iterations
$w_{i,n}^t$	the n-th parameter from client i in t iteration
$x_{i,n}^t$	the ranking of $w_{i,n}^t$ in w_n^t
A_n, B_n	the slope and intercept of repeated median linear regression
$e_{i,n}^t$	the normalised residual of the n-th parameter from client i in t iteration

does not update it throughout the training process, when new types of content emerge over time, the root data may become stale thereby harming the classifier's performance. Other recent studies employ spectral analysis [62], differential privacy [64], and deep model inspection [65] to guard against poisoning attacks, but, again, they do not use historical information to assess the reliability of clients. To measure client trustworthiness without collecting additional data at the server, in the next sections we show how to design a robust aggregation method to generate reputation automatically based on the historical behaviours of clients, which is a more realistic approach for a real FL-based decentralised system implemented as a browser extension for clients.

III. A ROBUST FL METHOD FOR CLASSIFYING SENSITIVE CONTENT ON THE WEB

In this section, we first show how to build an FL-based classifier for sensitive content. Then we design a reputation score for protecting against poisoning attacks. We analyse theoretically the combined FL/reputation-based solution and establish convergence and accuracy guarantees under common operating assumptions.

A. FL Framework for Classifying Sensitive Content

Table I presents the notation that we use in the remainder of the paper. In FL, clients provide the server updated parameters from their local model, which the server aggregates to build the global model M.

Suppose we have M clients participating in our classification training task and the dataset $\mathcal{D} = \bigcup_{i=1}^{M} \mathcal{D}_i$, where $\mathcal{D}_i \sim \mathcal{X}_i(\mu_i, \sigma_i^2)$ denotes the local data of client i from non-independent and non-identically (Non-IID) distribution $\mathcal{X}_i$ with the mean μ_i and standard deviation σ_i. In our task, the clients' data is the textual content of URLs stripped of HTML tags. The objective function of FL, $\mathcal{L} : \mathbb{R}^d \to \mathbb{R}$ which is the negative log likelihood loss in our task, can be described as

$$\mathcal{L}(w) = \mathbb{E}_{D \sim \mathcal{X}} \left[l(w; \mathcal{D}) \right]$$

where $l(w; \mathcal{D})$ is the cost function of parameter $w \in \mathcal{W} \subseteq \mathbb{R}^d$. Here we assume $\mathcal{W}$ is a compact convex domain with diameter d. Therefore, the task becomes

$$w^* = \underset{w \in \mathcal{W}}{\arg\min} \, \mathcal{L}(w)$$

To find the optimal w^*, we employ Stochastic Gradient Descent (SGD) to optimise the objective function.

During the broadcast phase, the server broadcasts the classification task and training instructions to clients. Then, the clients apply the following standard pre-processing steps on the webpage content, that is, transformation of all letters in lowercase and the removal of stop words. Next, the
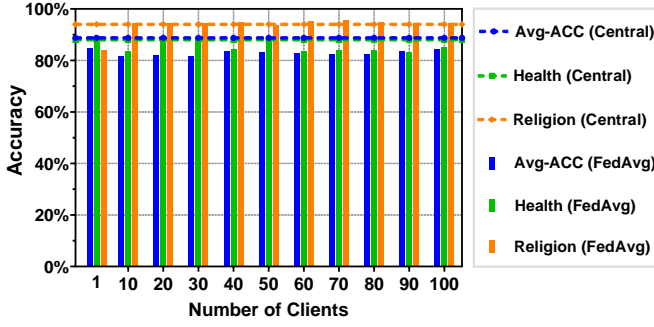
Fig. 1: Accuracy of FL classifiers and centralised classifiers in Health, Religion and all category.

clients extract the top one thousand features utilising the Term Frequency-Inverse Document Frequency (TF-IDF) [57] as in [4]. At iteration t, the client i receives the current global model M_{global} and then following the training instructions from server, trains the local model on its training data $\mathcal{D}_i$ and optimises $w_i^t = \arg\min_w \mathcal{L}_i(w_i^t)$ by using SGD:

$$w_i^t \leftarrow w_i^{t-1} - r\frac{\partial \mathcal{L}_i(w_i^{t-1})}{\partial w}$$

where $\mathcal{L}_i(w) := \mathbb{E}_{\mathcal{D}_i \sim \mathcal{X}}[l(w; \mathcal{D}_i)] = \frac{1}{Q_i}\sum_{j=1}^{Q_i} l(w; \mathcal{D}_i^j)$, $\mathcal{D}_i^j$ and Q_i means the j-th sample and the number of samples of the client i respectively, and r is the learning rate.

In every iteration, after finishing the training process the clients send back their local updates to the server. Then, the server computes a new global model update by combining the local model updates via an aggregation method AGG as follows:

$$w^t = \text{AGG}\left(\left\{w_i^t\right\}_{i=1}^M\right)$$

Here we utilise the basic aggregation method (FedAvg) [5], which uses the fraction of each client's training sample size in total training samples as the average weights:

$$w^t = \sum_{i=1}^M \frac{Q_i}{Q} w_i^t$$

We introduce other robust aggregation methods in the next subsection. Subsequently, the server uses the global model update to renew the global model M_{global}.

Using the above FL-based framework we first evaluate how the number of users in the system affects the average accuracy of the classifier. The results for the sensitive categories, Health and Religion, as well as the overall average accuracy (Avg-ACC) are depicted in Figure 1. A first observation is that when a fixed size dataset is divided into multiple segments and distributed to more clients, the model's accuracy decreases since each client has less data for training. Compared to the centralised classifier, using the same data, the accuracy of the FL classifier is slightly lower, which is expected when the training is distributed to a larger number of clients. Overall, we observe that the average accuracy difference between the FL and the centralised classifier is 5.76%, and this remains steady as the number of clients grows. In addition, Looking at the different sensitive categories (Health and Religion), we see the FL-based classifier achieves an accuracy very close to the corresponding one of the centralised classifier for these categories (on average 0.8533 vs. 0.88 and 0.9366 vs. 0.94, respectively).
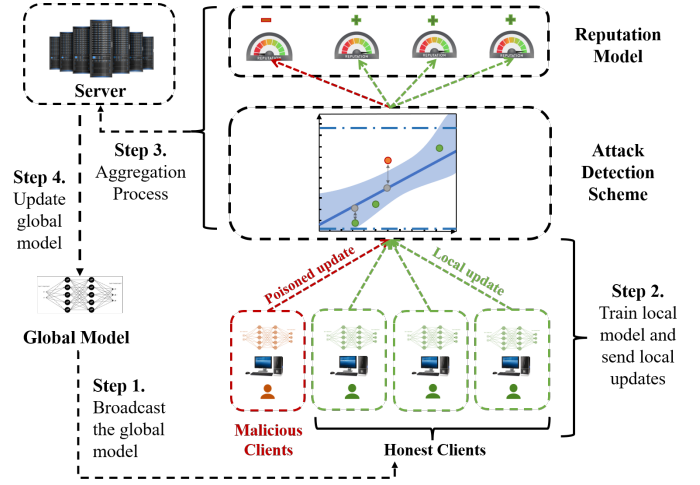

Fig. 2: Overview of reputation-based aggregation algorithm.

B. A Reputation score for Thwarting Poisoning Attacks

Figure 2 shows an overview of our reputation-based aggregation algorithm consisting of three components: the *attack detection scheme*, the *reputation model*, and the *aggregation module*. The attack detection scheme re-scales and rectifies damaging updates received from clients. Then, the reputation model calculates each client's reputation based on their past detection results. Finally, the aggregation module computes the global model by averaging the updates of the clients using their reputation scores as weights. We detail each component in the following subsections.

1) Attack Detection Scheme: Our attack detection scheme aims to reduce the impact of suspicious updates by identifying them and applying a rescaling algorithm. At every iteration, when model updates from clients arrive at the server, we apply Algorithm 1 there to rescale the range of values for those parameters in the updates.

This restriction on the value range aims not only to minimise the impact of abnormal updates from attackers but also to limit the slope for the repeat median regression. Considering the n-th parameter in round t from all the participants, we calculate the standard deviation $\sigma(w_{i,n}^t)$ of this series. Then we sort them in ascending order and determine the range by subtracting the lowest value from the highest one. If the result is above the threshold ϖ, we rescale the highest and lowest value by deducting and adding its standard deviation respectively to further bound their range.

Then, a robust regression [13] is carried out to identify outliers among the updates in the current round. Outlier detection is a well-established topic in statistics. Robust regression methods often handle outliers by using the median estimators. Median-based aggregation methods have a rich and longstanding history in the area of robust statistics [20]. However, the methods developed by the traditional robust statistics can only withstand a small fraction of Byzantine clients, resulting in a low "breakdown point" [58]. Different from many other variations of the univariate median, the repeated median [13] is impervious to atypical points even when their percentage is nearly 50%. The repeated median is defined as a modified U-statistic and the concept behind it is to utilise a succession of partial medians for computing approximation $\hat{\tau}$ of the

Algorithm 1: Rescale(w)

Input : $\{w_{i,n}^t\} \leftarrow$ Local Model parameters in round t

Output: $\{w_{i,n}^t\}$ with range of value less than ϖ

1 **for** $n \leftarrow 1$ **to** N **do**
2   // Determine the maximum range
3    $Rm = \max w_{i,n}^t - \min w_{i,n}^t = w_{i,n}^{t,(Max)} - w_{i,n}^{t,(Min)}$
4   **while** $Rm > \varpi$ **do**
5   // Rescale range based on standard deviation.
6       $w_{i,n}^{t,(Max)} := w_{i,n}^{t,(Max)} - \sigma(w_{i,n}^t)$;
7       $w_{i,n}^{t,(Min)} := w_{i,n}^{t,(Min)} + \sigma(w_{i,n}^t)$;
8   // Updated Rm.
9       $Rm = \max w_{i,n}^t - \min w_{i,n}^t = w_{i,n}^{t,(Max)} - w_{i,n}^{t,(Min)}$
10   **end while**
11 **end for**

---

parameter τ: For $k \in \mathbb{N}$, the value of parameter $\tau(z_1, \cdots, z_k)$ is determines by subset of k data points $z_1, \cdots, z_k$.

$$\hat{\tau} = \underset{z_1}{\text{median}} \left\{ \underset{z_2 \notin \{z_1\}}{\text{median}} \left\{ \underset{z_k \notin \{z_1, \cdots, z_{k-1}\}}{\text{median}} \tau(z_1, \cdots, z_k) \right\} \right\} \quad (1)$$

In our case, the intercept $\hat{A}$ and slop $\hat{B}$ are estimated by repeated median as bellow:

$$\hat{B}_n = \underset{i}{\text{median}} \left\{ \underset{i \neq j}{\text{median}} \{B_n(i,j)\} \right\} \quad (2)$$

$$\hat{A}_n = \underset{i}{\text{median}} \left\{ w_{i,n} - \hat{B}_n x_{i,n} \right\} \quad (3)$$

where $B_n(i,j) = \frac{w_{j,n} - w_{i,n}}{x_{j,n} - x_{i,n}}$, $x_{i,n}$ represents the index of $w_{i,n}$ in w_n which is sorted in ascending order.

Next, we employ the IRLS scheme [9] to generate each parameter's confidence score $s_{i,n}^t$ based on the normalised residual from repeated median regression, which is also utilised in a residual-based aggregation method [15]:

$$s_{i,n}^t = \frac{\sqrt{1 - diag(H_n^t)}}{e_{i,n}^t} \Psi\left(\frac{e_{i,n}^t}{\sqrt{1 - diag(H_n^t)}} \right) \quad (4)$$

where confidence interval $\Psi(x)$:

$$\Psi(x) = \max\{-\lambda\sqrt{2/M}, \min(\lambda\sqrt{2/M}, x)\}$$

and the hat matrix H_n^t:

$$H_n^t = x_n^t (x_n^{t^T} x_n^t)^{-1} x_n^{t^T}$$

with $e_{i,n}^t = \frac{25(M-1)\left(w_{i,n}^t - \hat{B}_n x_{i,n}^t - \hat{A}_n\right)}{37(M+4)\underset{i}{\text{median}}\left(|w_{i,n}^t - \hat{B}_n x_{i,n}^t - \hat{A}_n|\right)}$.

The distance between the point and the robust line is described by the confidence score derived from the normalised residual, which can be used to evaluate if the point is anomalous. Following the computation of the parameter's confidence score, and in light of the fact that some attackers want to generate updates with abnormal magnitudes in order to boost the damage, a useful protection is to identify low confidence values based on a threshold δ. Once the server recognises an update $\mathbf{w}_{i,n}^t$ of the client i with confidence values less than δ,

rather than altering this update to the repeat median estimation, our technique replaces it with the median of $\mathbf{w}_n^t$, as follows:

$$\mathbf{w}_{i,n}^t = \begin{cases} \mathbf{w}_{i,n}^t & \text{if } s_{i,n}^t > \delta \\ \underset{i}{\text{median}} \left\{ \mathbf{w}_{i,n}^t \right\} & \text{if } s_{i,n}^t \leq \delta \end{cases} \quad (5)$$

With the above, not only we bound the range of updates, but also improve the aggregation by introducing a robustness estimator.

2) Reputation Model: During the aggregation phase in FL, we use a subjective logic model to produce client reputation scores. The subjective logic model is a subset of probabilistic logic that depicts probability values of belief and disbelief as degrees of uncertainty [3]. In the subjective logic model, reputation score R_i^t for client i in t iteration correlates to a subjective belief in the dependability of the client's behaviour [38], as measured by the belief metric opinion τ_i^t [8]. An opinion is comprised of three elements: belief b_i^t, disbelief d_i^t and uncertainty u_i^t, with restrictions that $b_i^t + d_i^t + u_i^t = 1$ and $b_i^t, d_i^t, u_i^t \in [0,1]$. The reputation score may be calculated as the expected value of an opinion $E(\tau_i^t)$ which can be regarded as the degree of trustworthiness in client i. As a result, the value of the client's reputation is defined as follows:

$$R_i^t = E(\tau_i^t) = b_i^t + au_i^t \quad (6)$$

where $a \in [0,1]$ denotes the prior probability in the absence of belief, which reflects the fraction of uncertainty that may be converted to belief. On the other side, distinct observations determined by the rectification phase in our Algorithm 2 are used to count belief, disbelief, and uncertainty opinions. The positive observation denoted by P_i^t indicates that the update $\mathbf{w}_{i,n}^t$ is accepted ($s_{i,n}^t > \delta$), whereas a negative observation denoted by N_i^t indicates that the update is rejected ($s_{i,n}^t \leq \delta$). As a consequence, the positive observations boost the client's reputation, and vice versa. To penalise the negative observations from the unreliable updates, a higher weight η is assigned to negative observations than the weight κ to positive observations with constrain $\eta + \kappa = 1$. Therefore, in Beta distribution below:

$$Beta(p|\alpha, \beta) = \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha-1}(1-x)^{\beta-1} \quad (7)$$

with the constraints $0 \leq x \leq 1$, parameters $\alpha > 0, \beta > 0$, and $x \neq 0$ if $\alpha < 1$ and $x \neq 1$ if $\beta < 1$. The parameters α and β that represent positive and negative observations respectively, can be expressed as below

$$\begin{cases} \alpha = \kappa P_i^t + Wa \\ \beta = \eta N_i^t + W(1-a) \end{cases} \quad (8)$$

where W is the non-information prior weight and the default value is 2 [3].

As a consequence, the expected value of Beta distribution, which also stands for reputation value, can be calculated as follows:

$$E(Beta(p|\alpha, \beta)) = \frac{\alpha}{\alpha + \beta} = \frac{\kappa P_i^t + Wa}{\kappa P_i^t + \eta N_i^t + W} = R_i^t \quad (9)$$

Based on (6) and (9), we can derive

$$\begin{cases} b_i^t = \frac{\kappa P_i^t}{\kappa P_i^t + \eta N_i^t + W} \\ d_i^t = \frac{\eta N_i^t}{\kappa P_i^t + \eta N_i^t + W} \\ u_i^t = \frac{W}{\kappa P_i^t + \eta N_i^t + W} \end{cases} \quad (10)$$
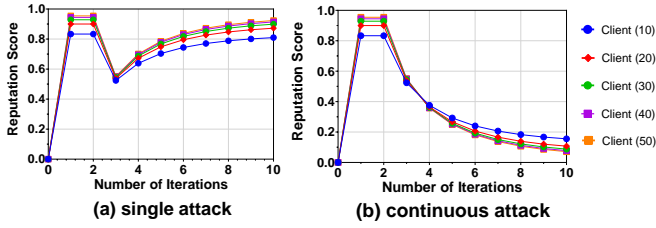
Fig. 3: The decay of reputation score in Client (X) with X model parameters when they (a) attack once at 3rd iteration and (b) attack continuously at and after the 3rd iteration.
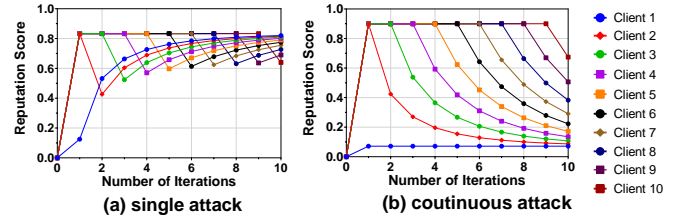

Fig. 4: The decay of reputation score in Client X with same model parameters when they (a) attack once at X iteration and (b) attack continuously after starting to attack at X iteration.

In addition, in order to take the client's historical reputation values in previous rounds into consideration, a time decay mechanism is included to lower the relevance of past performances without disregarding their influence. In other words, the reputation value from the most recent iteration contributes the most to the reputation model. We use exponential time decay in our model, as shown below:

$$\theta_{j,t} = \exp(-c(t-j)) \tag{11}$$

where $\exists c > 0$, $j \in [\tilde{s}, t]$, $\tilde{s} = \max(t-s, 0)$. We include a sliding window with a window length s that allows us to get a reputation for a certain time interval rather than the entire training procedure. We remove expired tuples with timestamps outside the window period during computation since they cannot provide meaningful information for the reputation. Hence, the final reputation score $\tilde{R}_i^t$ can be expressed as:

$$\tilde{R}_i^t = \frac{\sum_{j=\tilde{s}}^{t} \theta_{j,t} R_i^j}{\sum_{j=\tilde{s}}^{t} \theta_{j,t}} \tag{12}$$

To demonstrate how the reputation model evolves, we consider four scenarios where each client: (i) only attacks once at the same iteration, (ii) attacks continuously after launching an attack at the same iteration, (iii) only attacks once at different iteration, (iv) attacks continuously after launching an attack at different iteration. Here, clients conduct attacks as described in Section IV-A2 by utilising polluted data while training the local model, whereas the server uses our attack detection mechanism to identify these attacks.

Figure 3 displays the first two scenarios (i)-Figure 3a and (ii)-Figure 3b, respectively with Client X, who has X parameters in their local models, under single and continuous attack. In Figure 3a, all of the clients only attack once at the third iteration. When they start attacking, their reputation score plummets dramatically. In both scenarios, we observe the client who has more parameters has a larger relative decline in reputation score. This is also compatible with Corollary 1 in the Section III-C, that is, increasing the number of parameters N in the global model results in a lower error rate.

Figure 4 shows the last two scenarios (iii) and (iv) respectively with clients, who have 20 parameters in their local models, under single and continuous attack. In Figure 4a, Client X only launches an attack at X iteration. We observe that only one attack would lead to at least a 25.11% relative decrease in reputation score. In Figure 4b, Client X launches an attack at X iteration and keeps attacking in the following iterations. We observe in the end that 80% of their reputation
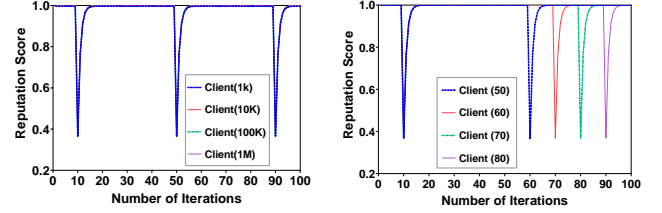

Fig. 5: The decay of reputation score in (left) Client with X model parameters when they attack at 10, 50 and 90 iteration; (right) Client X with 1 million parameters when they attack at 10 and $10 + X$ iteration.

scores are below 0.5, which is approximately half of the reputation score of honest clients, implying that the damage that they can inflict throughout the aggregation process is considerably decreased.

In addition, we consider a scenario in which an attacker spreads out the poisoning over a longer time duration, while using a higher number of model parameters. Figure 5 (left) depicts an attack over 40 iterations under different parameter sizes. Figure 5 (right) depicts an attack with 1 million parameters repeating every 50 to 80 iterations. These figures show that even if attackers spread our poisoning over multiple iterations and then try to recover their reputation score by acting benignly, our detection scheme can still identify them. This is because our attack detection and reputation schemes work in sequence. The attack detection scheme detects malicious updates without considering any reputation scores and rectifies them to mitigate damage. Then, the reputation scheme modifies the reputation scores based on the detection results. Also, attackers that employ a higher number of model parameters suffer a marginally higher reduction of reputation, which is consistent with Corollary 1.

3) Aggregation Algorithm: Algorithm 2 explains our aggregation method based on the attack detection scheme and subjective logic reputation model. First, the server sends all clients the pre-trained global model with initial parameters. Then, using their own data samples, clients train the global model locally and send the trained parameters back to the server. At this point, the server executes the attack detection scheme. In round t, if the n-th update parameter $\mathbf{w}_{i,n}^t$ from the client i has been rectified by the attack detection scheme in Section III-B1 to the median value, the server regards it as a negative observation, whereas no rectification represents a positive observation. Then, the server punishes the negative observation by reducing the corresponding client's reputation.

Algorithm 2: Aggregation Algorithm

Server :
Input : $w^0 \leftarrow$ Pretrained Model
$\quad\quad\quad \kappa,\eta,a,W,c,s \leftarrow$ Reputation parameters
Output: Global model M_{global} with w^T

```
 1 for Iteration t ← 1 to T do
 2   │ // Broadcast global model to clients
 3   │ send(w^{t-1});
 4   │ // Wait until all updates arrive
 5   │ receive(w^t);
 6   │ // Rescale parameters by Algorithm 1
 7   │ w̄^t ← Rescale(w^t);
 8   │ for n ← 1 to N do
 9   │   │ for i ← 1 to M do
10   │   │   │ // Compute parameter confidence
11   │   │   │ s_{i,n}^t = Eq 4(w̄_{i,n}^t);
12   │   │   │ // Rectify abnormal parameters
13   │   │   │ w_{i,n}^t := Eq 5 (s_{i,n}^t, δ);
14   │   │   │ record (P_i^t, N_i^t);
15   │   │ end for
16   │ end for
17   │ for i ← 1 to M do
18   │   │ // Calculate reputation score
19   │   │ R̃_i^t = Eq 12(P_i^t, N_i^t, κ, η, a, W, c, s);
20   │ end for
21   │ // Normalisation
22   │ R̄^t ← Norm(R̃^t);
23   │ for n ← 1 to N do
24   │   │ // Update the parameters
25   │   │ w_n^t := ∑_{i=1}^M (R̄_i^t / ∑_{i=1}^M R̄_i^t) w_{i,n}^t;
26   │ end for
27   │ // Obtain parameters for global model
28   │ w^t := [w_1^t, ···, w_n^t];
29 end for
```

Client :
```
 1 for Client i ← 1 to M do in parallel
 2   │ receive(w^{t-1});
 3   │ // Train local model
 4   │ w_i^t ← w_i^{t-1} - r (∂ℓ_i(w_i^{t-1}) / ∂w);
 5   │ send(w_i^t);
 6 end forpar
```

Both types of observations are accumulated through all the N parameters of client i to obtain the reputation value $\tilde{R}_i^t$ in t round for client i so as to all the other clients. The server would conduct Min-Max normalisation to obtain $\bar{R}^t$ after receiving the reputation values $\tilde{R}^t$ of all the clients in t round.

After the server gets correction updates and the normalised reputation of each client, it aggregates the updates using average weighted reputation as the weights to get our global model updates for the current iteration. In this way, even over many training rounds, the attackers are still incapable of shifting parameters notably from the target direction and this ensures the quality of the resulting global model as will be demonstrated experimentally and analytically next.

C. Theoretical Guarantees

We prove the convergence of our reputation-based aggregation method. Our major results are Theorem 1 and Corollary 1, which state that convergence is guaranteed in bounded time. Regarding the performance of our algorithm in terms of metric average accuracy and convergence, we show that it is consistent with our theoretical analysis. We start by stating our assumptions, which are standard and common for such types of results, and per recent works such as [7], [19].

Assumption 1 (Smoothness). *The loss functions are L-smooth, which means they are continuously differentiable and their gradients are Lipschitz-continuous with Lipschitz constant $L > 0$, whereas:*

$$\forall i \in N, \forall \mathbf{w}_1, \mathbf{w}_2 \in \mathbf{R}^d$$
$$\|\nabla\mathcal{L}(\mathbf{w}_1)) - \nabla\mathcal{L}(\mathbf{w}_2))\|_2 \leq L\|\mathbf{w}_1 - \mathbf{w}_2\|_2$$
$$\|\nabla\ell(\mathbf{w}_1; \mathcal{D}) - \nabla\ell(\mathbf{w}_2; \mathcal{D})\|_2 \leq L\|\mathbf{w}_1 - \mathbf{w}_2\|_2$$

Assumption 2 (Bounded Gradient). *The expected square norm of gradients $\sqsupseteq$ is bounded:*

$$\forall \mathbf{w} \in \mathbf{R}^d, \exists \mathcal{G}_\mathbf{w} < \infty, \mathbb{E}\|\nabla\ell(\mathbf{w}; \mathcal{D})\|_2^2 \leq \mathcal{G}_\mathbf{w}$$

Assumption 3 (Bounded Variance). *The variance of gradients $\mathbf{w}$ is bounded:*

$$\forall \mathbf{w} \in \mathbf{R}^d, \exists \mathcal{V}_\mathbf{w} < \infty, \mathbb{E}\|\nabla\ell(\mathbf{w}; \mathcal{D}) - \mathbb{E}(\nabla\ell(\mathbf{w}; \mathcal{D}))\|_2^2 \leq \mathcal{V}_\mathbf{w}$$

Assumption 4 (Convexity). *The loss function $\mathcal{L}(\sqsupseteq)$ are μ-strongly convex:*

$$\exists \mu > 0, \forall \mathbf{w}_1, \mathbf{w}_2 \in \mathbf{R}^d, \nabla\mathcal{L}(\mathbf{w}^*) = 0$$
$$\mathcal{L}(\mathbf{w}_1) - \mathcal{L}(\mathbf{w}_2) \geq \langle\nabla\mathcal{L}(\mathbf{w}_2), \mathbf{w}_1 - \mathbf{w}_2\rangle + \frac{\mu}{2}\|\mathbf{w}_1 - \mathbf{w}_2\|_2^2$$

Suppose the percentage of attackers in the whole clients is p, and all the clients in the system participant every training iteration. r is the learning rate($r < \frac{1}{L}$) and $\hat{Q} = \max\{Q_i\}_{i=1}^M$. $\forall \mathbf{w} \in \mathcal{W}$, we denote

$$\mathbf{m}_i(\mathbf{w}^t) = \begin{cases} * & \text{if } i \in \text{malicious clients} \\ \nabla l_i(\mathbf{w}^t; \mathcal{D}) & \text{if } i \in \text{honest clients} \end{cases}$$

where $*$ stands for an arbitrary value from the malicious clients.

$$\mathbf{m}(\mathbf{w}^t) = \sum_{i=1}^M \bar{R}_i \mathbf{m}_i(\mathbf{w}^t)$$

$$s.t. \ \bar{R}_i = \frac{\tilde{R}_i^t}{\sum_{i=1}^M \tilde{R}_i^t}, \ \sum_{i=1}^M \bar{R}_i = 1, \ \bar{R}_i \in (0,1)$$

Consider the assumptions above and lemmas presented in Appendix A, we have

Theorem 1. *Under Assumptions 1, 2, 3 and 4, $\exists \epsilon > 0$ that:*

$$\sqrt{\frac{d\log(1 + \hat{Q}MLD_\epsilon)}{M(1-p)}} + C\frac{\mathcal{G}_\mathbf{w}}{\sqrt{\hat{Q}}} + p \leq \frac{1}{2} - \epsilon \quad (13)$$

After t rounds, Algorithm 2 converges with probability at least $1 - \xi \in \left[1 - \frac{4d}{(1+\hat{Q}MLv)^d}, 1\right)$ as

$$\|\mathbf{w}^t - \mathbf{w}^*\|_2 \leq (1 - Lr)^t\|\mathbf{w}^0 - \mathbf{w}^*\|_2 + \frac{\sqrt{N}}{L}\Delta_1 + \frac{1}{L}\Delta_2 \quad (14)$$

where

$$\Delta_1 = \frac{M\left(\varpi(M-1) + \frac{2E}{\sqrt{M\delta}}\right)}{\frac{Wa(M-1)(\kappa N+W)}{(\eta N+W)(\kappa N+Wa)} + 1}$$

$$\Delta_2 = 2\sqrt{2}\frac{1}{M\hat{Q}} + \sqrt{\frac{2}{\hat{Q}}}D_\epsilon V_w \left(\sqrt{\frac{d\log(1+\hat{Q}MLv)}{M(1-p)}} + C\frac{\mathcal{G}_w}{\sqrt{\hat{Q}}} + p\right)$$

$$D_\epsilon := \sqrt{2\pi}\exp\left(\frac{1}{2}\left(\Phi(1-\epsilon)\right)^2\right)$$

with $\Phi(\cdot)$ being the cumulative distribution function of Wald distribution.

Corollary 1. *Continuing with Theorem 1, when the iterations satisfy $t \geq \frac{1}{Lr}\log\left(\frac{L}{\sqrt{N}\Delta_1+\Delta_2}\left\|\mathbf{w}^0 - \mathbf{w}^*\right\|_2\right)$, $\exists \xi \in \left(0, \frac{4d}{(1+\hat{Q}MLv)^d}\right]$, we have:*

$$\mathbb{P}\left(\left\|\mathbf{w}^t - \mathbf{w}^*\right\|_2 \leq \frac{2\sqrt{N}}{L}\Delta_1 + \frac{2}{L}\Delta_2\right) \geq 1 - \xi$$

Remark 1. *Due to*

$$\Delta_1 := \mathcal{O}\left(\frac{\varpi}{a\kappa WN} + \frac{1}{\kappa N} + \frac{1}{\sqrt{MN\delta}}\right)$$

and

$$\Delta_2 := \mathcal{O}\left(\frac{1}{\hat{Q}} + \frac{p}{\sqrt{\hat{Q}}} + \frac{1}{\sqrt{\hat{Q}M}}\right)$$

Based on Corollary 1, we achieve an error rate:

$$\mathcal{O}\left(\frac{\varpi}{a\kappa W\sqrt{N}} + \frac{1}{\kappa\sqrt{N}} + \frac{1}{\sqrt{M\delta}} + \frac{1}{\hat{Q}} + \frac{p}{\sqrt{\hat{Q}}} + \frac{1}{\sqrt{\hat{Q}M}}\right)$$

we observe the experimental results in Figure 3 and 11 of Sections III and IV respectively, when varying the parameters of N, p, a and κ, results are consistent with this error rate.

Remark 2. *Derived from the Corollary 1 and Remark 1, there is a trade-off problem between convergence speed and error rate according to the level of reward κ and punishment η from the reputation model. This trade-off problem is mainly based on the fact that if the model penalises the bad behaviours of clients heavily, it would decrease their reputation dramatically so the model would take a longer time to converge. On the other hand, mitigating the punishment to increase the reward, would lead to an increase in the error rate.*

IV. Performance Evaluation

The objectives of our experimental evaluation are the following: (a) evaluate the performance of our aggregation method against other state-of-art robust aggregation methods, (b) benchmark it in three different scenarios, namely, no attack, label flipping attack, and backdoor attack, (c) do so using a text based real-world dataset of sensitive categories from [4] to which we will henceforth refer to as SURL, and finally (d) show that our experimental result are consistent with our previous theoretical analysis.

A. Experimental Setup

1) Datasets: The SURL dataset comes from a crowd-sourcing taxonomy in the Curlie project [31], containing six categories of URLs: five sensitive categories (Health, Politics, Religion, Sexual Orientation, Ethnicity) and one for non-sensitive URLs, with a total of 442,190 webpages. The number of URLs in sensitive and non-sensitive categories are equally balanced. Each sample contains content, metadata and a class label of the webpage. For the SURL text classification task, we train a neural network with three fully connected layers and a final softmax output layer, same as in the evaluated methods [15], [19]. Furthermore, in order to fulfil the fundamental setting of an heterogeneous and unbalanced dataset for FL, we sample u_k from a Dirichlet distribution [17] with the concentration parameter $\iota = 0.9$ as in [11], which controls the imbalance level of the dataset, then assigns a $u_{k,i}$ fraction of samples in class k to client i, with the intention of generating non-IID and unbalanced data partitions. As a sanity check, we also tested our reputation scheme on a different classification task involving images and got consistent results as those we got for sensitive content (see Appendix C).

2) Threat Model: We consider the following threat model. **Attack capability:** In the FL setting, the malicious clients have complete control over their local training data, training process and training hyper-parameters, e.g., the learning rate, iterations and batch size. They can pollute the training data as well as the parameters of the trained model before submitting it to the server but cannot impact the training process of other clients. We follow the common practice in the computer security field of overrating the attacker's capability rather than underrating it, so we limit our analysis to worst-case scenarios. There, an attacker has perfect knowledge about the learning algorithm, the loss function, the training data and is able to inspect the global model parameters. However, attackers would still have to train with the model published by the server, thus complying with the prescribed training scheme by FL to their local data. Furthermore, the percentage of byzantine clients p is an important factor that determines the level of success for the attack. We assume that the number of attackers is less than the number of honest clients, which is a common setting in similar methods [15], [19] to the ones we evaluate and compare our method with.
Attack strategy: We focus on two common attack strategies for sensitive context classification, namely, (i) label flipping attack [23] and (ii) backdoor attack [11]. Comparing to other attacks, for example model poisoning attack [28], [62], these two data poisoning attacks are more likely to be carried out by real users in the real world via our browser extension described in Section V, since polluting data is easier than manipulating model updates using the browser extension. Note that privacy attacks including membership inference attack [64] and property inference attack [63], are out of the scope of this paper, but form part of our ongoing and future work.

In a label flipping attack, the attacker flips the labels of training samples to a targeted label and trains the model accordingly. In our case, the attacker changes the label of "Health" to "Non-sensitive". In a backdoor attack, attackers inject a designed pattern into their local data and train these manipulated data with clean data, in order to develop a local model that learns to recognise such pattern. We realise

backdoor attacks inserting the top 10 frequent words with their frequencies for the "Health" category. Therein the backdoor targets are the labels "non-sensitive". A successful backdoor attack would acquire a global model that predicts the backdoor target label for data along with specific patterns.

For both attacks, instead of a single-shot attack where an attacker only attacks in one round during the training, we enhance the attacker by a repeated attack schedule in which an attacker submits the malicious updates in every round of the training process. Also, we evaluate a looping attack where the attackers spreads out poisoning every 30 epochs for the label flipping attack based on Figure 5. It is important to note that even if the attackers have full knowledge of our method, they would still be unable to mount smarter attacks that would try to maximise the damage caused while minimising their reputation drop. This is because the attackers are unaware and cannot compute their reputation score since the latter is computed at the server and requires input from all clients. Moreover, we allow extra training epochs for an attacker, namely, being able to train the local models with 5 more epochs as in [11].

3) Evaluated Aggregation Methods: We compare the performance of our aggregation method against the existing state-of-the-art in the area FedAvg [5], as well as against popular robust aggregation methods such as Coordinate-wise median [19], Trimmed-mean [19], FoolsGold [14], Residual-based re-weighting [15] and FLTrust [59].

FedAvg is a FL aggregation method that demonstrates impressive empirical performance in non-adversarial settings [5]. Nevertheless, even a single adversarial client could control the global model in FedAvg easily [26]. This method averages local model updates of clients as a global model update weighted by the fraction of training samples size of each client compared to total training samples size. We use it as baseline evaluation to assess the performance of our method.
Median is using coordinate-wise median for aggregation. After receiving the updates in round t, the global update is set equal to the coordinate-wise median of the updates, where the median is the 1-dimensional median.
Trimmed-mean is another coordinate-wise mean aggregation technique that requires prior knowledge of the attacker fraction β, which should be less than half of the number of model parameters. For each model parameter, the server eliminates the highest and lowest β values from the updates before computing the aggregated mean with remaining values.
FoolsGold presents a strong defence against attacks in FL, based on a similarity metric. Such approach identifies attackers based on the similarity of the client updates and decreases the aggregate weights of participating parties that provide indistinguishable gradient updates frequently while keeping the weights of parties that offer distinct gradient updates. It is an effective defence for sybil attacks but it requires more iterations to converge to an acceptable accuracy.
Residual-based re-weighting weights each local model by accumulating the outcome of its residual-based parameter confidence multiplying the standard deviation of parameter based on the robust regression through all the parameters of this local model. In our reputation-based aggregation method, we implement the same re-weighting scheme IRLS [9] as residual-based aggregation, but choose the collection of reputation as the weights of clients' local models.

FLTrust establishes trust in the system by bootstrapping it via the server, instead of depending entirely on updates from clients, like the other methods do. The server obtains an initial server model trained on clean root data. Then, depending on the cosine similarity of the server model and each local model, it assigns a trust score to each client in each iteration.

4) Performance Metrics: We use the average accuracy (Avg-ACC) of the global model to evaluate the result of the aggregation defence for the poisoning attack in which attackers aim to mislead the global model during the testing phase. The accuracy is the percentage of testing examples with the correct predictions by the global model in the whole testing dataset, which is defined as:

$$\text{Avg-ACC} = \frac{\text{\# correct predictions}}{\text{\# testing samples}}$$

In addition, there is existence of targeted attacks that aim to attack a specific label while keeping the accuracy of classification on other labels unaltered. Therefore, instead of Avg-ACC, we choose the attack success rate (ASR) to measure how many of the samples that are attacked, are classified as the target label chosen by a malicious client, namely:

$$\text{ASR} = \frac{\text{\# successfully attacked samples}}{\text{\# attacked samples}}$$

A robust federated aggregation method would obtain higher Avg-ACC as well as a lower ASR under poisoning attacks. An ideal aggregation method can achieve 100% Avg-ACC and has the ASR as low as the fraction of attacked samples from the target label.

5) Evaluation Setup: For the malicious attack, we assume that 30% of the clients are malicious as in [26], which is also a common byzantine consensus threshold for resistance to failures in a typical distributed system [6]. For the server-side setting, in order to evaluate the reliability of the local model updates sent by the client to the server, we assume that the server has the ability to look into and verify the critical properties of the updates from the clients before aggregating.

Also, we only consider FL to be executed in a synchronous manner, as most existing FL defences require [7], [19], [26], [28]. For all the above aggregation methods under attack, we perform 100 iterations using the SURL dataset with a batch size of 64 and 10 clients. Furthermore, we evaluate our method for increasing numbers of clients. These settings are inline with existing state-of-the-art methods for security in FL [11], [15], [19] More details related to the training setting are presented in Appendix B.

B. Convergence and Accuracy

In Figure 6 left, we analyse the performance of our method in the no attack scenario and compare the convergence and accuracy of our method with others during training. We show the training loss (left axis) and average accuracy (right axis) during 100 training iterations for 7 methods.

Our aggregation starts with the lowest training loss and maintains it throughout the training process. It only takes 24 iterations to achieve 82% accuracy and then converge to 82.13%, which represents a $2.7\times$ faster converge rate than FedAvg. In comparison, Residual-based and Trimmed-mean have almost identical training loss and take 52 and 47 iterations
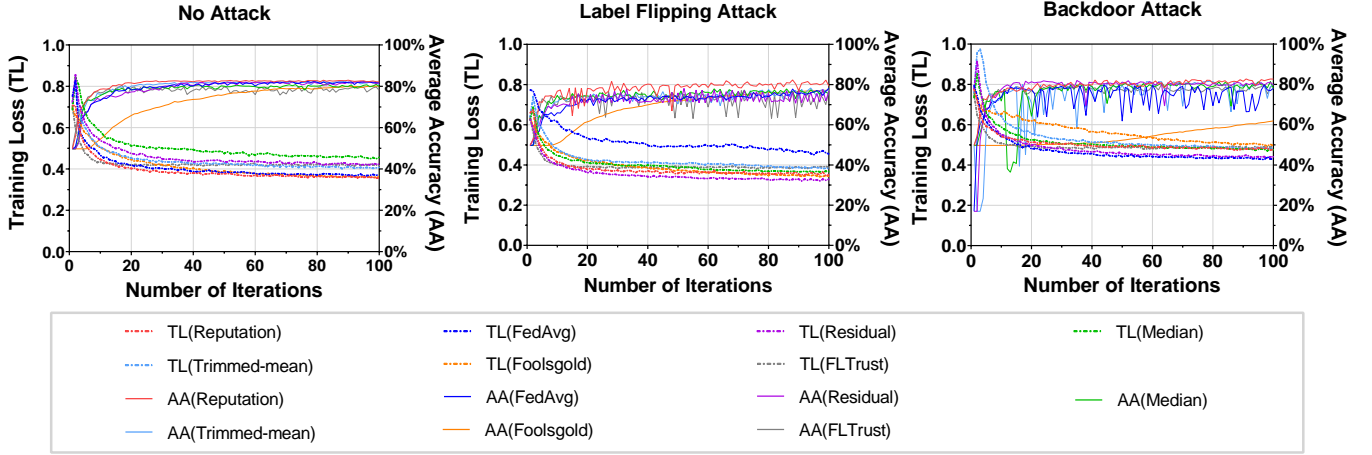
Fig. 6: Training Loss (TL) and Average Accuracy (AA) for 100 epochs of Reputation-based, FedAvg, Residual-based, Median, Trimmed-mean, FoolsGold and FLTrust methods in SURL Dataset under no attack (left) scenario, under label flipping attack (middle) and backdoor attack (right) scenarios with 30% malicious clients.

to reach 82% accuracy and practically converge to 81.79% and 80.76% respectively, which is 2.2× and 2× slower than our reputation method. Median reaches 81% at 83 rounds and after that converges to 79.94%, which amounts to a 3.6× slower converge rate than our method. Especially, Foolsgold and FLTrust are slow to converge and do not converge within 100 iterations, so our convergence rate is at least 4.2× better than Foolsgold and FLTrust. This demonstrates that our reputation model benefits from convergence speed and accuracy performance. This is because our reputation scheme assigns higher weight to more reliable clients when there is no ongoing attack, which generates more consistent updates thereby accelerating the convergence.

C. Resilience to Attacks

We begin by analysing the performance with a static percentage (30%) of attackers, and then move on to the performance with a varying percentage of attackers under label flipping and backdoor attacks.

1) Label Flipping Attack: **Static percentage of attackers:** Figure 6 (middle) shows the convergence of mentioned methods under label flipping attack. Our method converges 1.8× to at least 2× faster than all competing state-of-the-art methods under attack, enlarging its performance benefits compared to the no attack scenario. In addition, our method outperforms competing methods by at least 1.4% in terms of accuracy.
Varying the percentage of attackers: Here we analyse the impact on our aggregation method as the proportion of attackers increases. Figure 7 (left) shows the change of performance metrics for varying percentage of attackers for seven evaluated methods When the percentage of attackers p ranges from 10% to 50%, our method is resistant against label flipping attacks with a small loss in accuracy and a consistent attack success rate of all the methods. As p approaches 50%, FedAvg, Residual-based, Median and FLTrust defences become ineffective in mitigating the attack, and correspondingly their Avg-ACC decreases linearly. Moreover, under label flipping attack during the whole process, our reputation-based method has the highest accuracy outperforming other methods by 1%

to 23.1%. At the same time it has the lowest ASR; the average ASR of other methods are at least 82.8% higher than ours.

2) Backdoor Attack: **Static percentage of attackers:** Figure 6 (right) shows the convergence of mentioned methods under backdoor attack. Same as in no attack and label flipping attack scenario, our method converges 1.6× to at least 2.4× faster than all competing state-of-the-art methods. In addition, our method outperforms competing methods by 3.5% to 33.6% in terms of classification accuracy.
Varied percentage of attackers: We examine the scenario in which the percentage of attackers increases. Figure 7 (right) shows the performance for the seven evaluated methods under backdoor attack when varying the percentage of attackers p from 10% to 50%. Figure 7 (right) demonstrates that under backdoor attack, our reputation-based method has a consistent accuracy throughout the process with the lowest attack success rate, whereas the average ASR of other methods is at least 72.3% higher than ours. As p changes, the ASR of the Residual-based, Median, and Foolsgold methods increase linearly. Although FLTrust has a stable ASR, it increases by a factor of 1.39 when p reaches 50%.

First, we evaluate a varying compromise rate for the label flipping and backdoor attacks using our reputation-based method. For the label-flipping attack, we vary the percentage of the flipped label poisoned by attackers from 10% to 90%. Also, for the backdoor attack, we vary the number of top frequent words inserted as the trigger pattern, from 5 to 25. The remaining settings are the same as in previous experiments. Figure 8 plot the ACC and ASR when varying the compromise rate for both attacks. Figure 8 (left) shows that when the percentage of the poisoned sample is increased, it lead to the decrease of the accuracy of the model and to a slight increase of ASR. Figure 8 (right) shows that when we increase the number of frequent words from 5 to 20, the ASR remains unaffected. When the frequent words exceed 25, the attack becomes less stealthy and thus can be more easily detected resulting in a lower ASR.

We also evaluate the performance of our method in terms of the number of participating clients. With a 30% compromise rate, we expand the number of clients from 10 to 200. Our
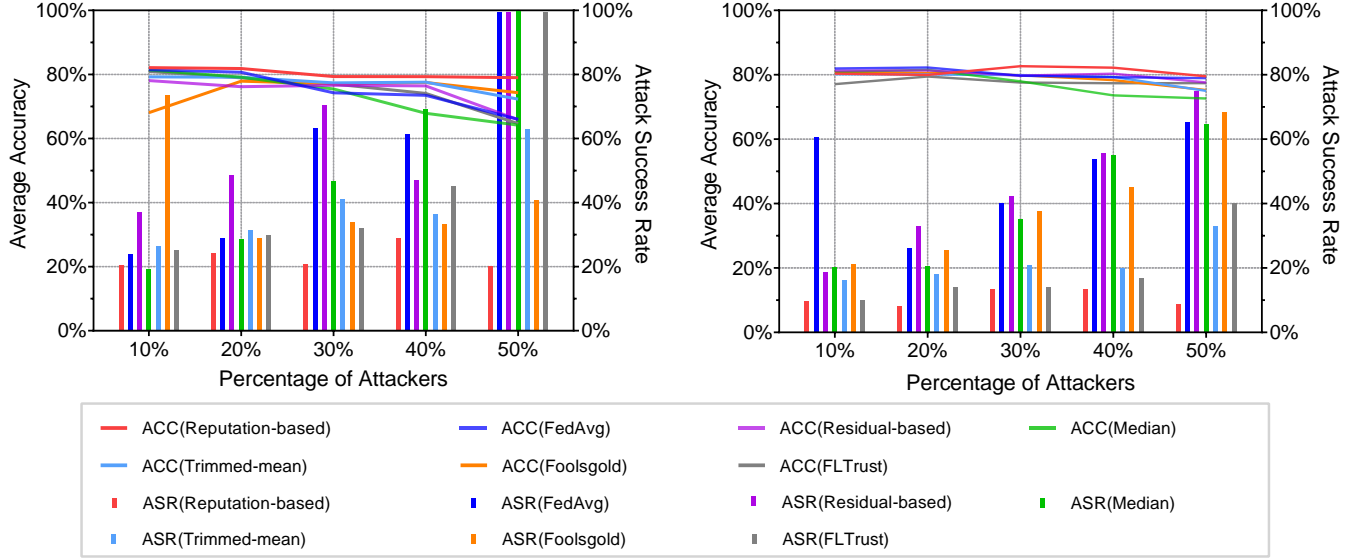
Fig. 7: Average accuracy (ACC) and attack success rate (ASR) for varying percentage of attackers from 10% to 50% under label flipping (left) and backdoor (right) attack for Reputation-based, FedAvg, Residual-based, Median, Trimmed-mean, FoolsGold and FLTrust methods in SURL Dataset.
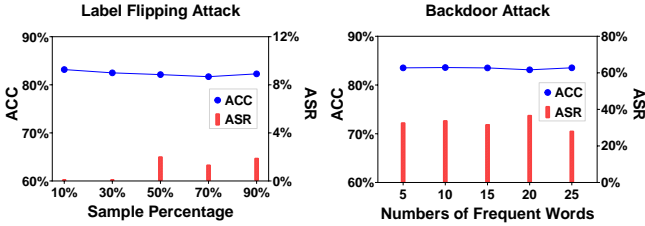

Fig. 8: ACC and ASR as we vary the percentage of flipped label from 10% to 90% (left) for label flipping attack, and the number of the frequent words as the trigger pattern from 5 to 25 for backdoor attack(right).
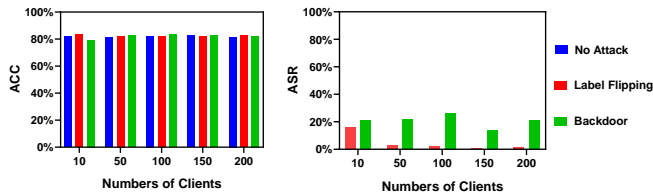

Fig. 9: Average accuracy (ACC) and attack success rate (ASR) for varying the number of clients from 10 to 200 under label flipping and backdoor attack with 30% malicious clients.

method performs consistently for a larger number of clients, as seen by the stable ACC and ASR as the number of clients grows in Figure 9.

3) Analysis of Attacks: Finally, instead of repeating the attack at every epoch, the attacker stretches poisoning across 30 epochs in our study of the looping attack. The performance of the looping attack is seen in Figure 10. As expected, the looping attack is not as effective as the repeated attack that we previously assessed. All the methods manage to defend it with low ASR, and our method still has the greatest accuracy.

4) Evaluation Results: In the no attack scenario, we observe (i) Our method converges $2\times$ to at least $4.2\times$ faster than all competing state-of-the-art methods. (ii) Our method is at least as good or outperforms competing methods in terms of classification accuracy. The above validates that our reputation scheme is helpful even in the no attack scenario. This is due to the fact that in our algorithm we give higher weights to the clients with high-quality updates, as illustrated in Figure 12, causing the model to converge rapidly and retain consistent accuracy. In addition, even under the two different attacks, our method:

- converges $1.6\times$ to at least $2.4\times$ faster than all competing state-of-the-art methods.
- provides the same or better accuracy than competing methods.
- yields the lowest ASR compared to all other methods, with the average ASR of them being at least 72.3% higher than ours.

We obtained comparable findings for the evaluation of the aforementioned methods on 100 clients, as presented in Appendix. Furthermore, the result is consistent with the theoretical analysis: as p increases, so does the error rate.

D. Stability of Hyper-parameters

We employ four hyper-parameters in our reputation model: rewarding weight κ, prior probability a, time decay parameter c and window length s. As shown in Remark 1, c and s do not affect the performance of our model, we only consider hyper-parameters κ and a, where κ controls the reward weight to positive observations and a controls the fraction of uncertainty converted to belief. To demonstrate the impact of these two hyper-parameters of our reputation model, we grid search κ in $[0.1, 0.2, 0.3, 0.4]$ and a in $[0.1, 0.3, 0.5, 0.7, 0.9]$. The setup is the same as on SURL dataset under label flipping attack. The ultimate accuracy of stability of reputation-based aggregation
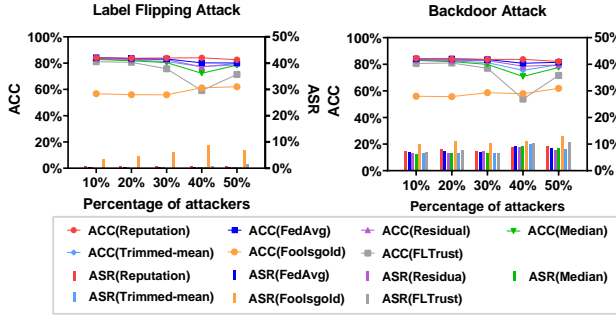
Fig. 10: Average accuracy (ACC) and attack success rate (ASR) for varying percentage of attackers from 10% to 50% under label flipping (left) and backdoor (right) attack with looping attack that attackers attack every 30 epochs.
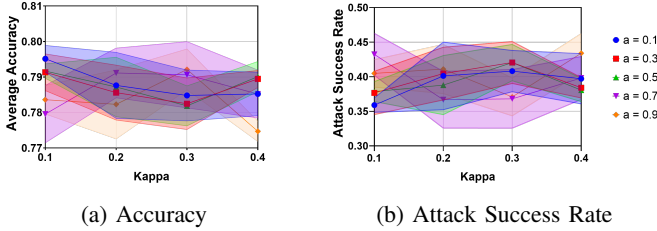

(a) Accuracy   (b) Attack Success Rate

Fig. 11: Average accuracy and attack success rate as we vary rewarding weight κ and prior probability a.

are shown in Figure 11. Note that these results are tested for the label flipping attack and they hold according to theory also for backdoor.

The result in Figure 11 demonstrates that our approach is very stable and efficient in terms of hyper-parameter selection, and it achieves a high degree of precision. Furthermore, the result is compatible with the theoretical analysis in Section III-C.

E. Comparison against a residual-based method

To demonstrate how our method improves the residual-base method by assigning the aggregation weights based on reputation, we consider a scenario with 10 clients in the FL system, 8 of which are malicious. The training lasts 10 communication rounds during which attackers carry out the backdoor attack. The remaining settings are the same as the default. Results are shown in Figure 12, in which the first two clients are benign, and the rest are malicious. We observe that for our reputation method the aggregation weights of malicious clients, which are their reputations, are rectified to 0 since the second round, demonstrating that our method is successful in eradicating their influence. On the other hand, the aggregate weights of malicious clients in residual-based methods, which are calculated by multiplying the parameter confidence by its standard deviation, are nearly similar and non-zero. This is because repeated median regression seldom yields 0 for the parameter confidence, which causes practically non-zero weights to be assigned to malicious clients by residual-based methods. To address this issue, the reputation model uses positive and negative observations that introduce rewards and punishments to assign divergent weights to clients. As a result, benign clients are given higher weights whereas malicious clients are eliminated from the aggregation.
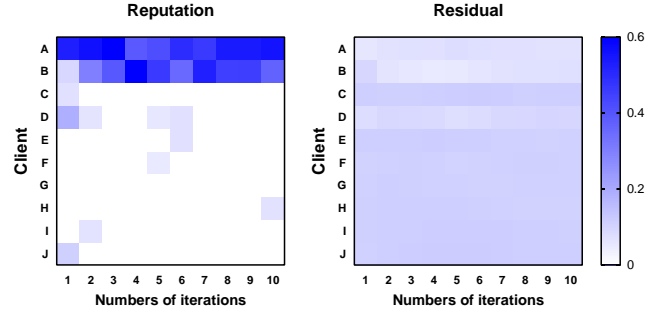

Fig. 12: The aggregation weights of clients from our reputation-based (left) and residual-based method (right) for 10 communication rounds under label flipping attack with 80% attackers.
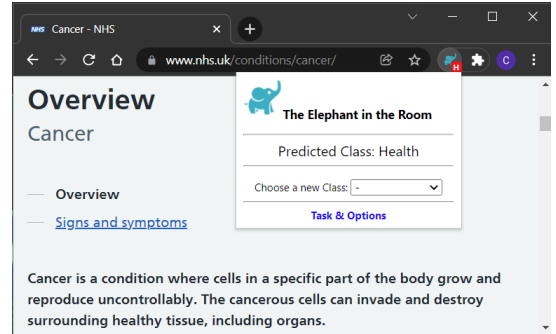

Fig. 13: *EITR* extension in action. The letter "H" inside the red frame at the bottom right of the extension's icon indicates that a health-related page has been detected.

V. THE *EITR* SYSTEM

In this section, we provide a high level description of our *EITR* system. We then present some preliminary results with real users demonstrating the ability of the system to quickly learn how to classify yet unseen sensitive content, in our case COVID-19 URLs pertaining to the category Health, even in view of inaccurate user input. The system is currently being used as a research prototype to evaluate the robustness of our algorithm in a simple real-world setting. A full in depth description of the system and its performance with more users and more intricate settings, including adoption, incentives, and HCI issues, over a longer time period is the topic of our ongoing efforts and will be covered by our future work.

A. System Architecture and Implementation

The *EITR* system is based on the client-server model. The back-end server is responsible to distribute the initial classification model and the consequent updated model(s) to the clients and receive new annotations from the different clients of the system. The client(s) is in the form of a web browser extension that is responsible to fetch and load the most recent global classification model to the users' browser from the back-end server. The loaded model can then be used to label website in real time into the 5 different sensitive topics as defined by GDPR, i.e., Religion, Health, Politics, Ethnicity and Sexual Orientation. Next, we provide more details for each part of the system.
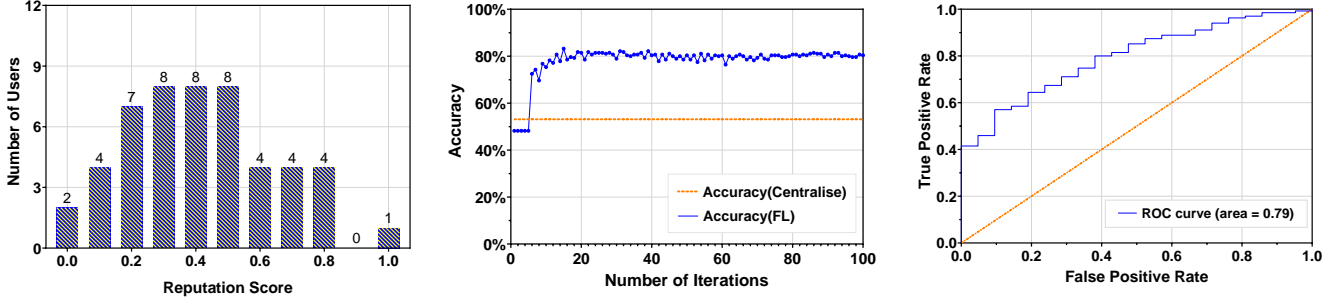
Fig. 14: Results of real-user experiment for COVID-19 related URLs with 50 users over 100 iterations. (left): the reputation score of the real users at the end of experiment, and (middle): the accuracy of the centralised classifier and the global model of our reputation-based FL approach for COVID-19 URLs, and (right): the ROC curve of real-user experiment with 0.79 area under the ROC curve (AUC).

Back-end server: The back-end server is written in JavaScript using the node.js Express [39] framework. To build the initial classification model we use the dataset provided by Matic et al. [4], and the TensorFlow [40] and Keras [41] machine learning libraries. The final trained model is then exported using the TensorFlow.js [42] library in order to be able to distribute it to the system's clients (browser extension). The back-end server also includes additional functionalities such as the creation and distribution of users' tasks – a short list of URLs that the users need to visit and annotate – and an entry point that collects the resulting users annotations during the execution of the task.

Web browser extension: Currently the browser extension only supports the Google Chrome browser and is implemented in JavaScript using the Google Chrome Extension APIs [44]. To handle the classification model the extension utilises the TensorFlow.js [42] library to load, use and update the model. The main functionality of the extension is to classify the visited website in real time and provide information to the user related to the predicted class as depicted in Figure 13. The website classification is based on the metadata (included in the website <head> HTML tag) and the visible text of the website. The extension also allows users to provide their input related to their agreement or disagreement with the predicted class using a drop down list as depicted in Figure 13 with the label *"Choose a new Class"*.

B. Real Users Experimental Setup

The goal of the real-user experiment is to evaluate our federated reputation-based method on real user activity (instead of systematic tests), and demonstrate that even with real users with different comprehensions of the definition of sensitive information, our method can learn new content fast and achieve higher accuracy than centralised classifiers, which is compatible with our simulation experiment.

For the setup, the participant is directed to visit the experiment website that provides the necessary information and instructions on the goal of our study, the definition of sensitive information provided by the current GDPR and how to participate in our study. In order to have access to the browser extension and the installation instructions the user must in advance give explicit consent and accept the data privacy policy. Upon successful installation of the extension, new users are asked to provide a valid email address (to contact

them for the reward) and then receive their task, a list of 20 URLs, that they need to visit and provide their labels in order to successfully complete their task. The list of 20 URLs is sampled by the Dirichlet distribution with $\iota = 0.9$ for each participant from a database, which includes 300 URLs with sensitive and non-sensitive content related to COVID-19.

Ethical Considerations: We have ensured compliance with the GDPR pertaining to collecting, handling and storing data generated by real users. To that end, we have acquired all the proper approvals from our institutions. Furthermore, the participants are directed to visit a pre-selected set of URLs selected by us to avoid collecting the actual visiting patterns of our users. In addition, the user input is only collected *if and only if* the user explicitly provide input to the drop-down list labelled *"Choose a new Class"* to avoid collecting the visiting patterns of the user accidentally while they are executing their tasks. Finally, we only use the users' email address to contact them for the reward. The mapping between the user input and their email address is based on a random identifier that is generated during the installation time of the extension.

C. Validation with Real Users

Data collection: We had 50 users participating in our experiment. In order to evaluate our reputation-based FL method using real-user data, we define a methodology to label ground truth on COVID-19 related sites.

Ground truth methodology: To set the ground truth for COVID-19 sites related to our sensitive or non-sensitive labels, we create a database of 100 websites, which we collect by searching on Google with the query "sensitive websites about COVID-19" and choose the top 100 sites returned from the query. Then, four experts in the privacy field, independently annotate them based on their professional expertise in order to achieve an agreement on whether each of those sites included sensitive or non-sensitive content.

Ground truth annotation: In order to evaluate the annotation of the 100 websites from human experts, we calculate the inter-rater agreement among them using Fleiss Kappa [60]. We obtain 0.56 of Fleiss Kappa, which is an acceptable agreement because the values of Fleiss Kappa. above 0.5 are regarded as good. Furthermore, given that COVID-19 is a controversial issue, it is difficult for humans to agree on what constitutes sensitive content relating to it. Even though, we still attain a valid ground truth of 85 items belonging to the health sensitive category with agreement ratings of at least 0.5. Note that

we also classify the above 100 websites using the centralised classifier proposed in [4] and get only 53.13% accuracy.

Result with real users: Figure 14 shows the results of accuracy and reputation score with 50 real users in the experiment. Figure 14 (left) shows that the majority of users have reputation scores falling in the intermediate range, with some having a very high reputation and a few having a very low reputation. This indicates the divergence of the user's interpretation of the sensitive information as we expect. In Figure 14 (middle) we compare the accuracy of the centralised classifier and the global model of our reputation-based FL approach for COVID-19 URLs. Despite the diversity of reputation scores of real users, our method converges as rapidly as in simulation and achieves an average accuracy of 80.36%, thereby verifying the quick convergence and high accuracy results presented in the previous sections. Figure 14 (right) shows that the ROC curve in real-user experiment yielded 0.79 AUC. Our result is acceptable in this scenario because most existing FL techniques are designed to minimise the conventional cost function and are not optimal for optimising more appropriate metrics for imbalanced data, such as AUC [61].

As we observe, with real users holding our method achieve a good performance. This means that, as new sensitive content appears and/or is defined by GDPR or new upcoming legislation, we will be able to continue training our FL model for this type of task with quick convergence and good accuracy. The empirical results in Figure 14 (middle) also shows that there is a quick convergence to the accuracy's stable value within a small number of iterations (around 30), in line with the theoretical results in Section III.

VI. Conclusion

In this paper we have shown how to use federated learning to implement a robust to poisoning attacks distributed classifier for sensitive web content. Having demonstrated the benefits of our approach in terms of convergence rate and accuracy against state-of-the-art approaches, we implemented and validated it with real users using our *EITR* browser extension. Collectively, our performance evaluation has showed that our reputation-based approach to thwarting poisoning attacks consistently converges faster than the state-of-the-art while maintaining or improving the classification accuracy.

We are currently working towards disseminating *EITR* to a larger user-base and using it to classify additional sensitive and non-sensitive types of content. This includes but it is not limited to categories defined by the users themselves for different purposes, not necessarily related to sensitive content, as well as evaluating additional attacks and threat models under our subjective-logic reputation scheme for FL. In turn, our approach can support other FL models going beyond sensitive content classification in future work.

References

[1] McMahan, H., Ramage, D., Talwar, K. & Zhang, L. Learning differentially private recurrent language models. *Proceedings of ICLR*. (2017)

[2] Mammen, P. Federated Learning: Opportunities and Challenges. *ArXiv Preprint ArXiv:2101.05428*. (2021)

[3] Jøsang, A. Subjective logic. (Springer,2016)

[4] Matic, S., Iordanou, C., Smaragdakis, G. & Laoutaris, N. Identifying Sensitive URLs at Web-Scale. *Proceedings Of The ACM Internet Measurement Conference*. pp. 619-633 (2020)

[5] McMahan, B., Moore, E., Ramage, D., Hampson, S. & Arcas, B. Communication-efficient learning of deep networks from decentralized data. *Artificial Intelligence And Statistics*. pp. 1273-1282 (2017)

[6] Castro, M., Liskov, B. & Others Practical byzantine fault tolerance. *OSDI*. **99**, 173-186 (1999)

[7] Xie, C., Koyejo, S. & Gupta, I. Zeno: Distributed stochastic gradient descent with suspicion-based fault-tolerance. *International Conference On Machine Learning*. pp. 6893-6901 (2019)

[8] Josang, A., Hayward, R. & Pope, S. Trust network analysis with subjective logic. *Conference Proceedings Of The Twenty-Ninth Australasian Computer Science Conference (ACSW 2006)*. pp. 85-94 (2006)

[9] Wilcox, R. Introduction to robust estimation and hypothesis testing. (Academic press,2011)

[10] Krizhevsky, A., Hinton, G. & Others Learning multiple layers of features from tiny images. (Citeseer,2009)

[11] Bagdasaryan, E., Veit, A., Hua, Y., Estrin, D. & Shmatikov, V. How to backdoor federated learning. *International Conference On Artificial Intelligence And Statistics*. pp. 2938-2948 (2020)

[12] Konečny, J., McMahan, H., Yu, F., Richtárik, P., Suresh, A. & Bacon, D. Federated learning: Strategies for improving communication efficiency. *Proceedings of NIPS*. (2016)

[13] Siegel, A. Robust regression using repeated medians. *Biometrika*. **69**, 242-244 (1982)

[14] Fung, C., Yoon, C. & Beschastnikh, I. Mitigating sybils in federated learning poisoning. *ArXiv Preprint ArXiv:1808.04866*. (2018)

[15] Fu, S., Xie, C., Li, B. & Chen, Q. Attack-resistant federated learning with residual-based reweighting. *The Workshop On Towards Robust, Secure And Efficient Machine Learning In The Thirty-Fifth AAAI Conference On Artificial Intelligence*. (2021)

[16] Sun, Z., Kairouz, P., Suresh, A. & McMahan, H. Can you really backdoor federated learning?. *The 2nd International Workshop On Federated Learning For Data Privacy And Confidentiality In Neural Information Processing Systems*. (2019)

[17] Minka, T. Estimating a Dirichlet distribution. (Technical report, MIT, 2000)

[18] He, K., Zhang, X., Ren, S. & Sun, J. Deep residual learning for image recognition. *Proceedings Of The IEEE Conference On Computer Vision And Pattern Recognition*. pp. 770-778 (2016)

[19] Yin, D., Chen, Y., Kannan, R. & Bartlett, P. Byzantine-robust distributed learning: Towards optimal statistical rates. *International Conference On Machine Learning*. pp. 5650-5659 (2018)

[20] Minsker, S. Geometric median and robust estimation in Banach spaces. *Bernoulli*. **21**, 2308-2335 (2015)

[21] Karimireddy, S., He, L. & Jaggi, M. Learning from history for byzantine robust optimization. *International Conference On Machine Learning*. pp. 5311-5319 (2021)

[22] Bubeck, S. Convex optimization: Algorithms and complexity. *Foundations and Trends in Machine Learning, 8 (3-4)*. (2014)

[23] Barreno, M., Nelson, B., Joseph, A. & Tygar, J. The security of machine learning. *Machine Learning*. **81**, 121-148 (2010)

[24] Deng, J., Dong, W., Socher, R., Li, L., Li, K. & Fei-Fei, L. Imagenet: A large-scale hierarchical image database. *2009 IEEE Conference On Computer Vision And Pattern Recognition*. pp. 248-255 (2009)

[25] Bhagoji, A., Chakraborty, S., Mittal, P. & Calo, S. Analyzing federated learning through an adversarial lens. *International Conference On Machine Learning*. pp. 634-643 (2019)

[26] Blanchard, P., El Mhamdi, E., Guerraoui, R. & Stainer, J. Machine learning with adversaries: Byzantine tolerant gradient descent. *Proceedings Of The 31st International Conference On Neural Information Processing Systems*. pp. 118-128 (2017)

[27] Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z.,

Lin, Z., Desmaison, A., Antiga, L. & Lerer, A. Automatic differentiation in pytorch. (2017)

[28] Fang, M., Cao, X., Jia, J. & Gong, N. Local model poisoning attacks to byzantine-robust federated learning. *29th USENIX Security Symposium*. pp. 1605-1622 (2020)

[29] Bonawitz, K., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H., Patel, S., Ramage, D., Segal, A. & Seth, K. Practical secure aggregation for privacy-preserving machine learning. *Proceedings Of The 2017 ACM SIGSAC Conference On Computer And Communications Security*. pp. 1175-1191 (2017)

[30] Guerraoui, R., Rouault, S. & Others The hidden vulnerability of distributed learning in byzantium. *International Conference On Machine Learning*. pp. 3521-3530 (2018)

[31] Curlie.org Curlie - The Collector of URLs. (2019), https://curlie.org/

[32] Commission, E. Data protection in the EU, The General Data Protection Regulation (GDPR); Regulation (EU) 2016/679. (2018), https://ec.europa.eu/%0Ainfo/law/law-topic/data-protection/

[33] California, S. California Consumer Privacy Act - Assembly Bill No. 375. (2018), https://leginfo.legislature.ca.gov/faces/billTextClient.xhtml?bill_id=201720180AB375

[34] Canada, O. Amended Act on The Personal Information Protection and Electronic Documents Act. (2018), https://www.priv.gc.ca/en/privacy-topics/privacy-laws-in-canada/the-personal-information-protection-and-electronic-documents-act-pipeda/

[35] Israel, T. Protection of privacy regulations (data security) 5777-2017. (2018), https://www.gov.il/en/Departments/legalInfo/data_%0Asecurity_regulation/

[36] Personal Information Protection Commission, J. Amended Act on the Protection of Personal Information. (2017), https://www.ppc.go.jp/en/

[37] Australian Information Commissioner, O. Australian Privacy Prin- ciples guidelines; Australian Privacy Principle 5 - Notification of the collection of personal information. (2018), https://www.oaic.gov.au/agencies-and-organisations/

[38] Kang, J., Xiong, Z., Niyato, D., Xie, S. & Zhang, J. Incentive mechanism for reliable federated learning: A joint optimization approach to combining reputation and contract theory. *IEEE Internet Of Things Journal*. **6**, 10700-10714 (2019)

[39] Expressjs.com Express - Fast, unopinionated, minimalist web framework for Node.js. (2021), https://expressjs.com/

[40] TensorFlow.org TensorFlow - An end-to-end open source machine learning platform.. (2021), https://www.tensorflow.org/

[41] Keras.io Keras - Simple. Flexible. Powerful.. (2021), https://keras.io/

[42] Tensorflow.org TensorFlow.js is a library for machine learning in JavaScript. (2021), https://www.tensorflow.org/js

[43] Bonawitz, K., Eichner, H., Grieskamp, W., Huba, D., Ingerman, A., Ivanov, V., Kiddon, C., Konečný, J., Mazzocchi, S., McMahan, H. & Others Towards federated learning at scale: System design. *Proceedings of MLSys*. (2019)

[44] Google Google Chrome Extension APIs. (2021), https://developer.chrome.com/docs/extensions/reference/

[45] Hard, A., Rao, K., Mathews, R., Ramaswamy, S., Beaufays, F., Augenstein, S., Eichner, H., Kiddon, C. & Ramage, D. Federated learning for mobile keyboard prediction. *ArXiv Preprint ArXiv:1811.03604*. (2018)

[46] Feng, J., Rong, C., Sun, F., Guo, D. & Li, Y. PMF: A privacy-preserving human mobility prediction framework via federated learning. *Proceedings Of The ACM On Interactive, Mobile, Wearable And Ubiquitous Technologies*. **4**, 1-21 (2020)

[47] Yu, T., Li, T., Sun, Y., Nanda, S., Smith, V., Sekar, V. & Seshan, S. Learning context-aware policies from multiple smart homes via federated multi-task learning. *2020 IEEE/ACM Fifth International Conference On Internet-of-Things Design And Implementation (IoTDI)*. pp. 104-115 (2020)

[48] Huang, L., Shea, A., Qian, H., Masurkar, A., Deng, H. & Liu, D. Patient clustering improves efficiency of federated machine learning to predict mortality and hospital stay time using distributed electronic medical records. *Journal Of Biomedical Informatics*. **99** pp. 103291 (2019)

[49] Brisimi, T., Chen, R., Mela, T., Olshevsky, A., Paschalidis, I. & Shi, W. Federated learning of predictive models from federated electronic health records. *International Journal Of Medical Informatics*. **112** pp. 59-67 (2018)

[50] Lee, J., Sun, J., Wang, F., Wang, S., Jun, C. & Jiang, X. Privacy-preserving patient similarity learning in a federated environment: development and analysis. *JMIR Medical Informatics*. **6**, e7744 (2018)

[51] Ahn, E., Kumar, A., Feng, D., Fulham, M. & Kim, J. Unsupervised deep transfer feature learning for medical image classification. *2019 IEEE 16th International Symposium On Biomedical Imaging (ISBI 2019)*. pp. 1915-1918 (2019)

[52] Lin, B., He, C., Zeng, Z., Wang, H., Huang, Y., Soltanolkotabi, M., Ren, X. & Avestimehr, S. FedNLP: Benchmarking Federated Learning Methods for Natural Language Processing Tasks. *Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*. (2022)

[53] Iordanou, C., Soriente, C., Sirivianos, M. & Laoutaris, N. Who is fiddling with prices? building and deploying a watchdog service for e-commerce. *Proceedings of ACM SIGCOMM* (2017)

[54] Iordanou, C., Kourtellis, N., Carrascosa, J., Soriente, C., Cuevas, R. & Laoutaris, N. Beyond content analysis: Detecting targeted ads via distributed counting. *Proceedings Of The 15th International Conference On Emerging Networking Experiments And Technologies*. pp. 110-122 (2019)

[55] Su, D., Cao, J., Li, N., Bertino, E. & Jin, H. Differentially private k-means clustering. *Proceedings Of The Sixth ACM Conference On Data And Application Security And Privacy*. pp. 26-37 (2016)

[56] Cormode, G. & Muthukrishnan, S. An improved data stream summary: the count-min sketch and its applications. *Journal Of Algorithms*. **55**, 58-75 (2005)

[57] Salton, G. & Buckley, C. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*. **24**, 513-523 (1988)

[58] Lopuhaa, H. & Rousseeuw, P. Breakdown points of affine equivariant estimators of multivariate location and covariance matrices. *The Annals Of Statistics*. pp. 229-248 (1991)

[59] Cao, X., Fang, M., Liu, J. & Gong, N. FLTrust: Byzantine-robust Federated Learning via Trust Bootstrapping. *Proceedings Of NDSS*. (2021)

[60] Fleiss, J. Measuring nominal scale agreement among many raters.. *Psychological Bulletin*. **76**, 378 (1971)

[61] Yuan, Z., Guo, Z., Xu, Y., Ying, Y. & Yang, T. Federated deep AUC maximization for hetergeneous data with a constant communication complexity. *International Conference On Machine Learning*. pp. 12219-12229 (2021)

[62] Shejwalkar, V. & Houmansadr, A. Manipulating the byzantine: Optimizing model poisoning attacks and defenses for federated learning. *NDSS*. (2021)

[63] Melis, L., Song, C., De Cristofaro, E. & Shmatikov, V. Exploiting Unintended Feature Leakage in Collaborative Learning. *2019 IEEE Symposium On Security And Privacy (SP)*. pp. 691-706 (2019)

[64] Naseri, M., Hayes, J. & De Cristofaro, E. Local and Central Differential Privacy forRobustness and Privacy in Federated Learning. *In Proceedings of NDSS*. (2022)

[65] Rieger, P., Nguyen, T., Miettinen, M. & Sadeghi, A. DeepSight: Mitigating Backdoor Attacks in Federated Learning Through Deep Model Inspection. *In Proceedings of NDSS*. (2022)

APPENDIX

A. Proofs

The following are the lemmas we utilise in the proof of Theorem 1.

Lemma 1. *From Assumption 1 and 4, $\mathcal{L}(w)$ is L-smooth and μ-strongly convex. Then $\forall w_1, w_2 \in \mathcal{W}$, one has*

$$\langle \nabla \mathcal{L}(\mathbf{w}_1) - \nabla \mathcal{L}(\mathbf{w}_2), \mathbf{w}_1 - \mathbf{w}_2 \rangle \geq \frac{L\mu}{L+\mu} \|\mathbf{w}_1 - \mathbf{w}_2\|_2^2$$
$$+ \frac{1}{L+\mu} \|\nabla \mathcal{L}(\mathbf{w}_1) - \nabla \mathcal{L}(\mathbf{w}_2)\|_2^2 \quad (15)$$

Lemma 2. *The difference between* $\mathbf{m}(\mathbf{w})$ *and* $\nabla\mathcal{L}(\mathbf{w})$ *is bounded in every iteration:*

$$\|\mathbf{m}(\mathbf{w}) - \nabla\mathcal{L}(\mathbf{w})\|_2 \leq \|\mathbf{m}_0(\mathbf{w}) - \nabla\mathcal{L}(\mathbf{w})\|_2 + \sqrt{N}\Delta_1 \quad (16)$$

where:

$$\Delta_1 = \frac{M(\varpi(M-1) + \frac{2E}{\sqrt{M}\delta})}{\frac{Wa(M-1)(\kappa N+W)}{(\eta N+W)(\kappa N+Wa)} + 1}$$

$$E = \sup\left\{\frac{37\sqrt{2}\lambda(M+4)}{25(M-1)}\operatorname*{median}_i\left\{|w_{i,n}^t - \hat{B}_n x_{i,n}^t - \hat{A}_n|\right\}\right\}$$

and

$$\mathbf{m}_0(\mathbf{w}) := \operatorname*{median}_i\{\mathbf{m}_i(\mathbf{w})\}$$

1) Proof of Lemma 1: Let $g(\mathbf{w}) = \mathcal{L}(\mathbf{w}) - \frac{\varsigma}{2}\|\mathbf{w}\|_2^2$. Base on the assumption 4, we have $g(\mathbf{w})$ is $(L-\varsigma)$-strongly convex. from [22] 3.6, we have

$$\langle\nabla\mathcal{L}(\mathbf{w}_1) - \nabla\mathcal{L}(\mathbf{w}_2), \mathbf{w}_1 - \mathbf{w}_2\rangle \geq \frac{1}{L}\|\nabla\mathcal{L}(\mathbf{w}_1) - \nabla\mathcal{L}(\mathbf{w}_2)\|_2^2 \quad (17)$$

Hence,

$$\langle\nabla g(\mathbf{w}_1) - \nabla g(\mathbf{w}_2), \mathbf{w}_1 - \mathbf{w}_2\rangle \geq \frac{1}{L-\varsigma}\|\nabla g(\mathbf{w}_1) - \nabla g(\mathbf{w}_2)\|_2^2 \quad (18)$$

Now We have

$$\langle\nabla\left(\mathcal{L}(\mathbf{w}_1) - \frac{\varsigma}{2}\|\mathbf{w}_1\|_2^2\right) - \nabla\left(\mathcal{L}(\mathbf{w}_2) - \frac{\varsigma}{2}\|\mathbf{w}_2\|_2^2\right), \mathbf{w}_1 - \mathbf{w}_2\rangle$$
$$\geq \frac{1}{L+\mu}\left\|\nabla\left(\mathcal{L}(\mathbf{w_1}) - \frac{\varsigma}{2}\|\mathbf{w}_1\|_2^2\right) - \nabla\left(\mathcal{L}(\mathbf{w}_2) - \frac{\varsigma}{2}\|\mathbf{w}_2\|_2^2\right)\right\|_2^2 \quad (19)$$

And therefore

$$\langle\nabla\mathcal{L}(\mathbf{w}_1) - \nabla\mathcal{L}(\mathbf{w}_2), \mathbf{w}_1 - \mathbf{w}_2\rangle - \langle\varsigma\mathbf{w}_1 - \varsigma\mathbf{w}_2, \mathbf{w}_1 - \mathbf{w}_2\rangle$$
$$\geq \frac{1}{L-\varsigma}\|(\nabla\mathcal{L}(\mathbf{w}_1) - \nabla\mathcal{L}(\mathbf{w}_2)) - (\varsigma\mathbf{w}_1 - \varsigma\mathbf{w}_2)\|_2^2 \quad (20)$$

Refer to Assumption 1, we obtain

$$\langle\nabla\mathcal{L}(\mathbf{w}_1) - \nabla\mathcal{L}(\mathbf{w}_2), \mathbf{w}_1 - \mathbf{w}_2\rangle \geq \frac{L\varsigma}{L-\varsigma}\|\mathbf{w}_1 - \mathbf{w}_2\|_2^2$$
$$- \frac{2\varsigma}{L-\varsigma}\langle\nabla\mathcal{L}(\mathbf{w}_1) - \nabla\mathcal{L}(\mathbf{w}_2), \mathbf{w}_1 - \mathbf{w}_2\rangle$$
$$+ \frac{1}{L-\varsigma}\|\nabla\mathcal{L}(\mathbf{w}_1) - \nabla\mathcal{L}(\mathbf{w}_2)\|_2^2$$
$$\geq -\frac{L\varsigma}{L-\varsigma}\|\mathbf{w}_1 - \mathbf{w}_2\|_2^2 + \frac{1}{L-\varsigma}\|\nabla\mathcal{L}(\mathbf{w}_1) - \nabla\mathcal{L}(\mathbf{w}_2)\|_2^2 \quad (21)$$

Let $\varsigma = -\mu$, then we conclude the proof of Lemma 1.

2) Proof of Lemma 2: We have the following equation:

$$\|\mathbf{m}(\mathbf{w}) - \nabla\mathcal{L}(\mathbf{w})\|_2 \leq \|\mathbf{m}(\mathbf{w}) - \mathbf{m}_0(\mathbf{w})\|_2$$
$$+ \|\mathbf{m}_0(\mathbf{w}) - \nabla\mathcal{L}(\mathbf{w})\|_2 \quad (22)$$

from [15] inequality 18, we know $\forall i, n, \exists E > 0$

$$\sup|e_{i,n}| \leq \frac{E}{\sqrt{M}\delta}$$

Where

$$E = \sup\left\{\frac{37\sqrt{2}\lambda(M+4)}{25(M-1)}\operatorname*{median}_i\left\{|w_{i,n}^t - \hat{B}_n x_{i,n}^t - \hat{A}_n|\right\}\right\}$$

and the dimension of $\mathbf{w}$ is N. Hence the distance between the two aggregation functions satisfies

$$\|\mathbf{m}(\mathbf{w}) - \mathbf{m}_0(\mathbf{w})\|_2 \leq \sqrt{N}\left\|\sum_{i=1}^M \bar{R}_i\left(\hat{B}_i(M-1) + \frac{2E}{\sqrt{M}\delta}\right)\right\|_2 \quad (23)$$

Based on Equation 12

$$\frac{s\exp(-cs)}{\sum_{j=0}^s \exp(-cj)} \cdot \frac{Wa}{\eta N + W} \leq \tilde{R}_i^t \quad (24)$$

$$\tilde{R}_i^t \leq \frac{s}{\sum_{j=0}^s \exp(-cj)} \cdot \frac{\kappa N + Wa}{\kappa N + W} \quad (25)$$

so we have

$$\bar{R}_i = \frac{\tilde{R}_i^t}{\sum_{i=1}^M \tilde{R}_i^t} \leq \frac{1}{\frac{Wa(M-1)(\kappa N+W)}{(\eta N+W)(\kappa N+Wa)} + 1} \quad (26)$$

Due to our Aggregation Algorithm

$$\hat{B}_n = \operatorname*{median}_i\left(\operatorname*{median}_{i\neq j}\frac{w_{j,n} - w_{i,n}}{x_{j,n} - x_{i,n}}\right) \leq \varpi \quad (27)$$

Therefore, we have

$$\left\|\sum_{i=1}^M \bar{R}_i\left(\hat{B}_i(M-1) + \frac{2E}{\sqrt{M}\delta}\right)\right\|_2 \leq \frac{M(\varpi(M-1) + \frac{2E}{\sqrt{M}\delta})}{\frac{Wa(M-1)(\kappa N+W)}{(\eta N+W)(\kappa N+Wa)} + 1}$$
$$= \Delta_1 \quad (28)$$

Hence, we proof Lemma 2.

3) Proof of Theorem 1: We first consider a general problem of robust estimation of a one dimensional random variable. Suppose that there are M clients, and p percentage of them are malicious and own adversarial data. In t iteration, we have:

$$\|\mathbf{w}^t - \mathbf{w}^*\|_2 = \|(\mathbf{w}^{t-1} - r\mathbf{m}(\mathbf{w}^{t-1}) - \mathbf{w}^*\|_2$$
$$\leq \underbrace{\|\mathbf{w}^{t-1} - r\nabla\mathcal{L}(\mathbf{w}^{t-1}) - \mathbf{w}^*\|_2}_{A}$$
$$+ \underbrace{r\|\mathbf{m}(\mathbf{w}^{t-1}) - \nabla\mathcal{L}(\mathbf{w}^{t-1})\|_2}_{B} \quad (29)$$

We bound part A first. We have

$$\|\mathbf{w}^{t-1} - r\nabla\mathcal{L}(\mathbf{w}^{t-1}) - \mathbf{w}^*\|_2^2 = \|\mathbf{w}^{t-1} - \mathbf{w}^*\|_2^2$$
$$+ r^2\|\nabla\mathcal{L}(\mathbf{w}^{t-1})\|_2^2 - 2r\langle\nabla\mathcal{L}(\mathbf{w}^{t-1}), \mathbf{w}^{t-1} - \mathbf{w}^*\rangle \quad (30)$$

Under the Assumption 4 and Lemma 1, we have

$$\langle \nabla \mathcal{L}(\mathbf{w}^{t-1}), \mathbf{w}^{t-1} - \mathbf{w}^* \rangle \geq \frac{L\mu}{L+\mu} \left\| \mathbf{w}^{t-1} - \mathbf{w}^* \right\|_2^2$$
$$+ \frac{1}{L+\mu} \left\| \nabla \mathcal{L}(\mathbf{w}^{t-1}) \right\|_2^2 \quad (31)$$

Then we combine inequalities 31 to equation 30

$$\left\| \mathbf{w}^{t-1} - r\nabla \mathcal{L}(\mathbf{w}^{t-1}) - \mathbf{w}^* \right\|_2^2 \leq (1 - 2r\frac{L\mu}{L+\mu}) \left\| \mathbf{w}^{t-1} - \mathbf{w}^* \right\|_2^2$$
$$+ (r^2 - \frac{2r}{L+\mu}) \left\| \nabla \mathcal{L}(\mathbf{w}^{t-1}) \right\|_2^2$$
$$(32)$$

From Assumption 1, we can derive:

$$\left\| \nabla \mathcal{L}(\mathbf{w}^{t-1}) - \nabla \mathcal{L}(\mathbf{w}^*) \right\|_2^2 \leq L^2 \left\| \mathbf{w}^{t-1} - \mathbf{w}^* \right\|_2^2 \quad (33)$$

Combining inequalities 32 and 33, we have:

$$\left\| \mathbf{w}^{t-1} - r\nabla \mathcal{L}(\mathbf{w}^{t-1}) - \mathbf{w}^* \right\|_2^2 \leq (1 - Lr)^2 \left\| \mathbf{w}^{t-1} - \mathbf{w}^* \right\|_2^2 \quad (34)$$

Let $r < \frac{1}{L}$, we have

$$\left\| \mathbf{w}^{t-1} - r\nabla \mathcal{L}(\mathbf{w}^{t-1}) - \mathbf{w}^* \right\|_2 \leq (1 - Lr) \left\| \mathbf{w}^{t-1} - \mathbf{w}^* \right\|_2 \quad (35)$$

Then we turn to bound part B. Based on Lemma 2, we know:

$$\left\| \mathbf{m}(\mathbf{w}) - \nabla \mathcal{L}(\mathbf{w}) \right\|_2 \leq \left\| \mathbf{m}_0(\mathbf{w}) - \nabla \mathcal{L}(\mathbf{w}) \right\|_2 + \sqrt{N}\Delta_1 \quad (36)$$

Assume Assumption 1, 2, 3 and 4 holds, and $\exists \epsilon$ fulfills inequality 13. Based on Lemma 1 in [19], with the probability $1 - \xi \geq 1 - \frac{4d}{(1+\hat{Q}MLv)^d}$, we have

$$\left\| \mathbf{m}_0(\mathbf{w}) - \nabla \mathcal{L}(\mathbf{w}) \right\|_2 \leq \sqrt{\frac{2}{\hat{Q}}} D_\epsilon V_w (\sqrt{\frac{d\log(1+\hat{Q}MLv)}{M(1-p)}}$$
$$+ C\frac{\mathcal{G}_w}{\sqrt{\hat{Q}}} + p) + 2\sqrt{2}\frac{1}{M\hat{Q}} = \Delta_2 \quad (37)$$

where $C = 0.4748$. After obtaining the bound of part A and B, now we have

$$\left\| \mathbf{w}^t - \mathbf{w}^* \right\|_2 \leq \underbrace{(1 - Lr) \left\| \mathbf{w}^{t-1} - \mathbf{w}^* \right\|_2}_{Bound\ A} + \underbrace{r\left(\sqrt{N}\Delta_1 + \Delta_2\right)}_{Bound\ B}$$
$$(38)$$

Hence, we can prove Theorem 1 through iterations using the formula of a finite geometric series,

$$\left\| \mathbf{w}^t - \mathbf{w}^* \right\|_2 \leq (1 - Lr)^t \left\| \mathbf{w}^0 - \mathbf{w}^* \right\|_2$$
$$+ \frac{1 - (1 - Lr)^t}{Lr} r\left(\sqrt{N}\Delta_1 + \Delta_2\right)$$
$$\leq (1 - Lr)^t \left\| \mathbf{w}^0 - \mathbf{w}^* \right\|_2 + \frac{1}{L}\left(\sqrt{N}\Delta_1 + \Delta_2\right)$$
$$(39)$$

B. Experimental Setting

Our simulation experiments are implemented with Pytorch framework [27] on the cloud computing platform Google Colaboratory Pro (Colab Pro) with access to Nvidia K80s, T4s, P4s and P100s with 25 GB of Random Access Memory.

Table II shows the default setting in our experiments.

TABLE II: Default experimental settings

Explanation	Notation	Default Setting
prior probability	a	0.5
non-information prior weight	W	2
weight for positive observation	κ	0.3
time decay parameter	c	0.5
window length	s	10
confidence threshold	δ	0.1
value range	ϖ	2
Objective Function	$\mathcal{L}(\cdot)$	Negative Log-likelihood Loss
Learning rate	r	0.01
Batch size per client		64
The number of local iterations		10
The number of total iterations		100

C. Supplementary dataset for experiment

CIFAR-10 is a supplementary dataset assessing the robustness of our reputation scheme in image datasets to poisoning attacks. The CIFAR-10 dataset is a 32×32 colour image dataset that includes ten classes with a total number of 50k images for training and 10k images for testing. Here we use ResNet-18 [18] model pertaining to ImageNet [24] with 20 iterations. In CIFAR-10 dataset, attackers switch the label of "cat" images to the "dog".

Figure 15 shows that under label flipping and backdoor attack during the whole process, our reputation-based method has the highest accuracy outperforming other methods and the lowest ASR, excluding Foolsgold in CIFAR-10, yielding a result that is similar to the one in SURL.

D. Extra Experimental Result

We evaluate the proposed defence for a varying number of clients from 10 to 200 in the SURL dataset. Here, we analysis the performance of the aforementioned methods for 100 clients case. The results are depicted in Figure 16 and Figure 17 that correspond to the average accuracy (ACC) and attach success rate (ASR), respectively.

In the no attack scenario, we observe our method converges $1.7\times$ to at least $7.1\times$ faster than all competing state-of-the-art methods with at least as good or outperforms competing methods in terms of classification accuracy, see Figure 16 (left). In addition, even under the two different attacks, our method: (i) converges $1.6\times$ to at least $3.6\times$ faster than all competing state-of-the-art methods, (ii) provides the same or better accuracy than competing methods, and (iii) yields the lowest ASR compared to all other methods.
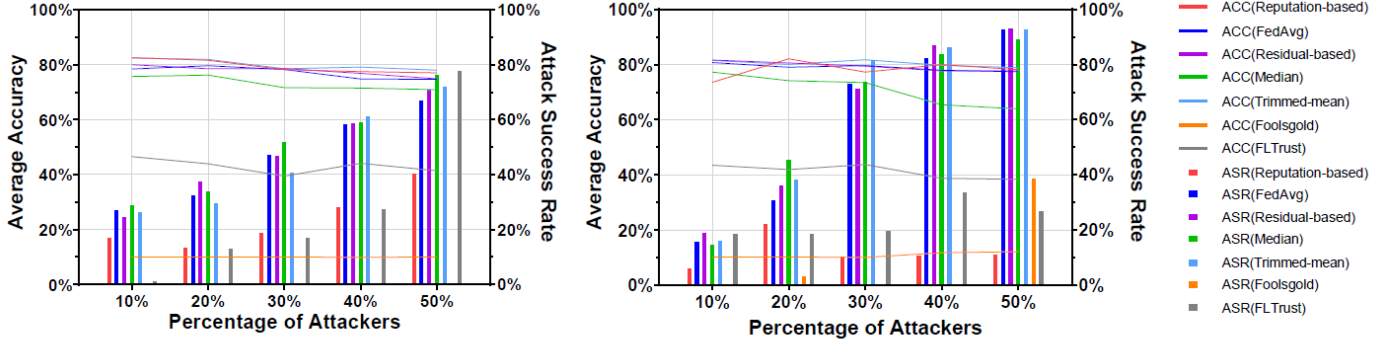
Fig. 15: Average accuracy (ACC) and attack success rate (ASR) for varying percentage of attackers from 10% to 50% under label flipping (left) and backdoor (right) attack for Reputation, FedAvg, Median and Residual in CIFAR-10 dataset.
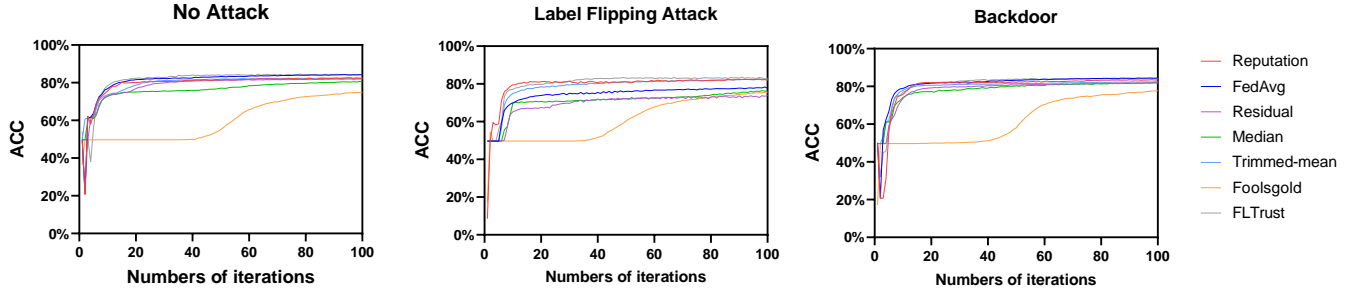

Fig. 16: Average accuracy (ACC) with no attack (left) and varying percentage of attackers from 10% to 50% under label flipping (middle) and backdoor (right) attack for Reputation, FedAvg, Residual-based, Median, Trimmed-mean, Foolsgold, FLTrust in SURL dataset.
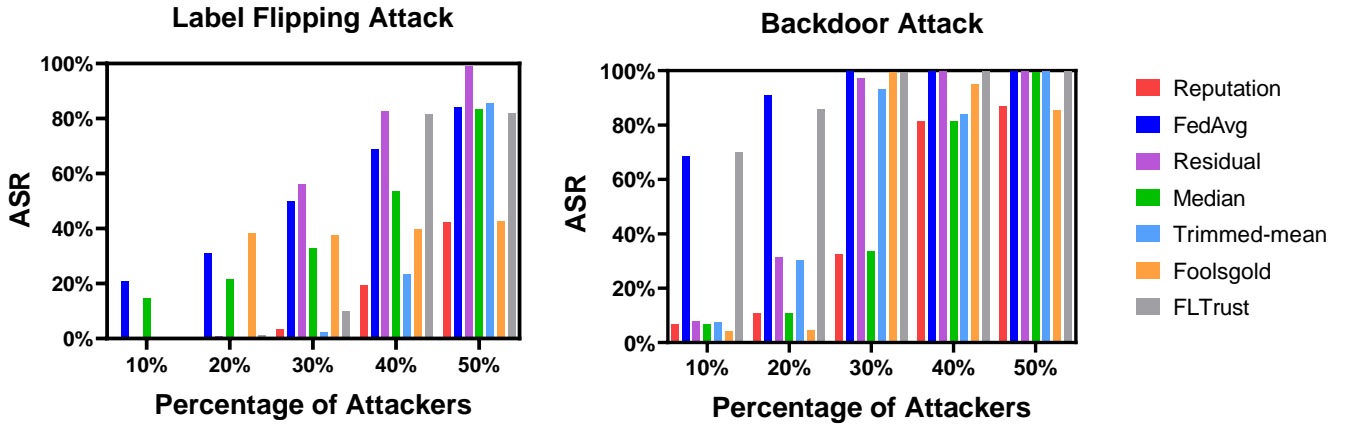

Fig. 17: Attack success rate (ASR) for varying percentage of attackers from 10% to 50% under label flipping (left) and backdoor (right) attack for Reputation, FedAvg, Residual-based, Median, Trimmed-mean, Foolsgold, FLTrust in SURL dataset.