Rec-Def: A Recommendation-Based Defence Mechanism for Privacy Preservation in Federated Learning Systems

Chamara Sandeepa, *Student Member, IEEE*, Bartlomiej Siniarski, *Member, IEEE*, Shen Wang, *Senior Member, IEEE*, and Madhusanka Liyanage, *Senior Member, IEEE*

***Abstract*—An emergence of attention and regulations on consumer privacy can be observed over the recent years with the ubiquitous availability of IoT systems handling personal data. Federated Learning (FL) arises as a privacy-preserved Machine Learning (ML) technique where data can be kept private within these devices without transmitting to third parties. Yet, many privacy attacks against FL can still leak private and sensitive information. Though solutions are currently available, they may not fit into the scope of IoT devices and may lead to sub-optimal results for the FL process. To balance these trade-offs, we propose a consumer-first approach to privacy protection where local privacy preservation is done via a privacy recommendation system. To evaluate the level of vulnerability of the local FL models, we use existing attacks and propose a novel time-based inference attack to test the resilience of FL models. Based on the vulnerability assessment, privacy recommendations are applied to local FL models based on a new gradient-split mechanism that adds a perturbation mask to the updates. Our experiments demonstrate the attacks can be mitigated effectively via the proposed mechanisms, with enhanced privacy and minimum compromise to the FL model utility.**

***Index Terms*—Federated learning, privacy, inference, defence techniques, privacy recommender, communication networks.**

I. Introduction

With the increased demand for IoT-based consumer devices and the development of rapid communication of these devices over 5G and beyond networks, the exponential growth of big data can be observed in recent years. However, it also creates a critical potential for the leakage of user privacy if data is directly sent to a third party. As a technique to overcome this, FL has emerged over recent years to facilitate distributed ML while preserving privacy. In consumer devices, FL allows training ML models within the device from local user data without transmitting data to a third party.

However, recent research on FL-based attacks [1], [2], [3] show it is possible to infer end-user data from the shared ML models, which can still cause leakages in privacy. These privacy leakages occur when forwarding the local model parameters to the aggregator. A malicious aggregator or an eavesdropper can inspect these forwarded model parameters. Then, they can perform privacy attacks from the captured parameters. Thus, this paper discusses two key privacy leakages in FL: membership inference attacks and data reconstruction via deep leakage from gradient attacks. Membership inference attacks aim to infer if the target model is trained on a particular data record the attacker has already captured. As a novel contribution, we extend membership inference attacks by introducing a new concept when performed continuously over multiple FL rounds, and we call them Federated Learning Time Inference Attacks (FL-TIA). Unlike the original information intended in membership inference or data reconstruction, FL-TIA aims to identify changes in the dataset over time, such as inferring significant events in the training data over time. In the experiments, we also launch the other attack, deep leakage, that intends to recover the original client datasets.

To illustrate the impact of the attacks in a real-world scenario, we select the use case of Virtual Reality (VR) headsets-based services in the metaverse. The metaverse is a virtual world where users can interact in real time with multi-dimensional data, creating a realistic perception. VR headsets play a crucial role as the key consumer IoT device, where they capture private and sensitive information such as user motion, facial expressions, and depth information. Metaverse service providers can provide FL-based services for VR headsets, which optimise the recognition and sensing capabilities of the device via FL models.

Privacy attacks can affect client-oriented services like facial expression recognition and device-based background intelligence services, such as network Intrusion Detection Systems (IDS). For example, suppose an attacker launches a deep leakage attack to recover the facial expression data of VR headset consumers. Then, they can reveal the user's identity through FL model updates. Furthermore, the attackers can identify user behaviour patterns upon inspection of changes in facial expressions. When considering IDS background services, our membership variation attack FL-TIA can be performed to identify the period when certain events, such as attacks, occurred in the device. Revealing such time-related information can let the attacker assess the security

Manuscript received 30 May 2023; revised 23 August 2023 and 9 October 2023; accepted 27 October 2023. Date of publication 16 November 2023; date of current version 26 April 2024. This work was supported in part by the European Union in SPATIAL under Grant 101021808, and in part by the Science Foundation Ireland through CONNECT Phase 2 Project under Grant 13/RC/2077_P2. *(Corresponding author: Chamara Sandeepa.)*

The authors are with the School of Computer Science, University College Dublin, Dublin 4, D04 V1W8 Ireland (e-mail: abeysinghe.sandeepa@ucdconnect.ie; bartlomiej.siniarski@ucd.ie; shen.wang@ucd.ie; madhusanka@ucd.ie).

Digital Object Identifier 10.1109/TCE.2023.3333560

vulnerabilities of the device, which they can exploit to gain access to the client device at more vulnerable periods in the network.

Adversarial ML techniques can be applied to provide resilience against such attack scenarios. Here, defences are based on mitigating factors like overfitting that can also lead to privacy attacks. One of the strategies to provide defence is through perturbation. This perturbation can be done during model training, where the original model is added with a controlled noise level. However, another problem occurs since the quantities of noise cannot be set as constants for each individual in the case of FL since each dataset can be different. Furthermore, a third party cannot inspect the dataset's properties, which would violate the key privacy idea in FL. Therefore, consumers who use FL-based services choose to either use sub-optimal privacy preservation mechanisms or risk their privacy by allowing third parties access to data properties. To overcome this issue, we propose a novel client-based local defence framework. It provides recommendations on the levels of privacy mechanisms on FL-based services in IoT devices.

Our contributions of this paper as follows:

- Introduce novel inference attack types through time and launch data reconstruction attacks for a use case of VR-based IoT consumer headsets.
- Propose a novel defence technique resilient against inference attacks and data reconstruction based on splitting the gradients and adding a random noise-based mask.
- Propose a new recommendation framework for privacy updates based on testing the resilience of the FL models via launching privacy attacks against it.
- Experimentally validate the significance and impact of the proposed approach and compare it with existing privacy preservation techniques.

The rest of the paper is arranged as follows. Section II provides existing related studies. Section III presents an overview of our system model. Details on the proposed attacks are presented in Section IV, and defence techniques are presented in Section V. Section VI discusses the experiments performed. In Section VII, we discuss the impact of the attacks, limitations, and potential future work. The paper concludes in Section VIII.

II. Related Works

A. Federated Learning for Recommender Systems

As FL can provide privacy-preserved ML capabilities for consumer devices, we observe a surge of applications in FL. They include use cases in multi-access edge computing, robot networking, consumer and industrial engineering applications, and healthcare [4]. Recommender systems-based applications can also get support from FL to improve their recommendation results while maintaining privacy. FL-driven recommender systems can address the issue of data silo issue where multiple parties are not willing to share data to train a common recommender system directly [5]. This can be helpful in launching applications like VR-based multisensory platforms, where multiple third-party services can collectively share their FL models and obtain better recommendations for users who use these services. The work in [6] demonstrates an online recommendation platform based on FL for multiple consumer-based applications such as product recommendations, online advertising, and content recommendations. However, recent works identify numerous types of threats for FL.

B. Privacy Attacks on FL

Many related works describe FL-based privacy attacks, including membership inference, determining class representatives, property inference on data, and model inversion attacks. The survey in [7] classifies different privacy leakage scenarios based on insiders, like malicious clients and servers, and outsiders, like consumers and eavesdroppers.

In addition, several new types of inference attacks are investigated in the research literature. The work in [1] proposes source inference, a technique to distinguish the client that added specific data for training the FL model. However, it does not discuss identifying training rounds where the data was entered. The authors in [2] discuss another type of inference attack called category inference, where an attacker attempts to determine the category information of the data used by the clients. They use a multi-train classifier inference model with an approximate model update technique. The work [8] uses property inference to identify the first appearance of specific features in training data; however, they may be applicable only for particular properties on the dataset and not for two data with the same property.

Authors in [9] use *shadow models* for improving the accuracy of membership inference attacks in ML models. A Generative Adversarial Network (GAN)-based membership inference attack on FL is proposed in [3]. An attacker can generate any amount of artificial data samples using a GAN model. A GAN consists of two components: a discriminator and a generator. Representative data is fed to the discriminator. The generator uses noise to generate new samples similar to the input dataset. Another type of privacy attack is called a model inversion attack, where the attacker's aim is to reconstruct the original dataset from the FL model gradients. Examples of such attacks include Deep Leakage from Gradients (DLG) [10] and improved Deep Leakage from Gradients (iDLG) [11], where the attacker uses dummy gradients and minimizes the loss of the gradients, meanwhile recovering the original dataset during the process.

C. Defence Techniques

Several defence mechanisms have been introduced to combat privacy leakage and attacks in FL. One common method is to use Differential Privacy (DP) [12], where an artificial noise can be added to the FL model at the client side before aggregating. This method can provide quantifiable privacy bounds for adding the noise, which provides a guarantee of privacy. However, with higher privacy, the model utility gets reduced due to increasing noise quantities added [12].

Another mechanism for defending against privacy leakage is Multiparty Computation (MPC), where FL model updates are split into secret shares among multiple clients distributed in the network, and they jointly compute partial aggregations without

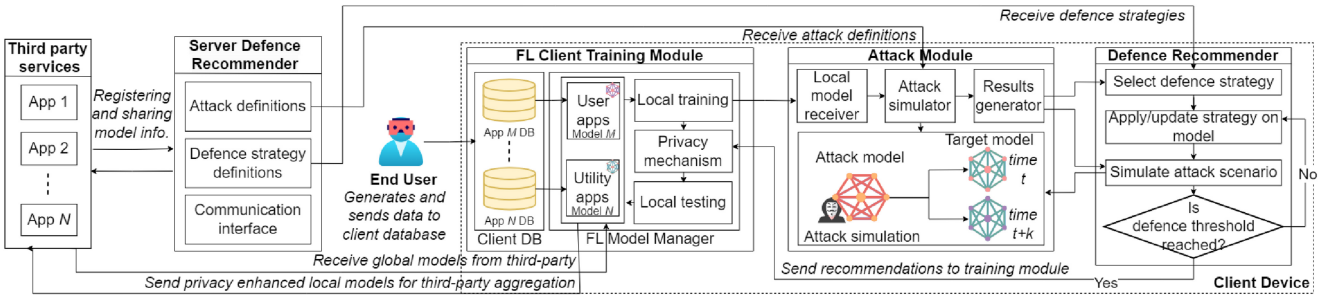


Fig. 1. Overview of the FL Rec-Def framework.

revealing individual model details [13]. This method can maintain the utility of the models since no noise is required to be added. However, it may increase the communication overhead among clients in the network [13]. It could be costly for IoT devices such as VR, which require providing real-time virtual experience with low latency. Homomorphic Encryption (HE) is another privacy mechanism that can perform aggregations over encrypted models [14]. However, ciphertext generated from HE tends to require much larger storage, which also increases the communication overhead [15].

When considering the existing privacy frameworks for FL, in [16], the authors present a privacy framework for FL by applying perturbation and dummy model updates from the client side and multiparty computation from the server side. They show the reduction of model accuracy with higher noise levels. Work in [17] proposes a blockchain-based secure FL framework for 5G networks. They use local DP and Gaussian noise to mitigate membership inference attacks. However, with the increase in privacy, the accuracy of the models also got reduced due to DP-injected noise. Similarly, authors in [18] design a blockchain and smart contract-based FL framework, where only outsider attacks are mitigated and do not provide resilience against adversarial insiders.

Therefore, we investigate a lightweight FL defence mechanism that can maintain the model's utility over iterations at no extra bandwidth cost. The defence we provide can be implemented on consumer IoT devices to provide recommendations by integrating it into our novel privacy framework.

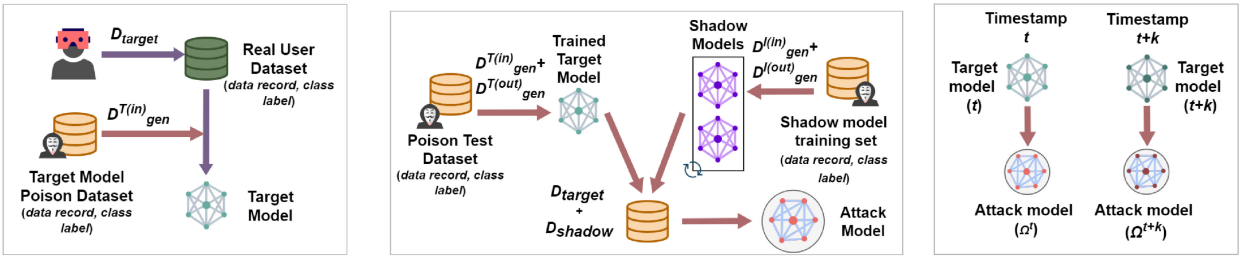
III. Proposed Rec-Def Privacy Framework

Privacy is a major concern for consumer devices in the case of IoT systems. The majority of present works focus on server-side security and privacy of such systems. An example of IoT-based DDoS defence is proposed in [19]. It provides resilience against DDoS attacks for multi-layer IoT-DDoS and, therefore, is suited for metaverse server-side applications. However, in the case of consumer privacy, individuals are the most vulnerable to privacy attacks. Therefore, we propose a consumer-first type defence recommender framework named *Rec-Def* against FL privacy attacks. It is placed on the client side, where clients perform privacy updates based on the recommendations to the FL models before forwarding them to a third-party server. We use VR headsets as the use case of IoT devices in the system. In practice, multiple VR devices may have different underlying implementations on core components, yet they all should support third-party services like apps and Software as a Service (SaaS) interfacing. This framework is specifically designed for such types of services where these third parties are untrustworthy. Though multiple devices have different configurations, they all can access the same framework via a common Application Programming Interface (API) service. Therefore, it is easier to scale the recommendation system to support many VR device types.

The proposed framework consists of three main components in the client: 1) the FL client training module, which primarily works on training local models that are specific to either user-based or utility-based services; 2) the attack module, which performs resilience testing via launching privacy attacks on the models; and 3) FL defence recommender module, that recommends local FL model defence techniques to mitigate the attack. Fig. 1 provides an overview of the framework.

The metaverse VR applications provide FL services, where local training is performed with the models shared by the aggregation server. We consider two types of ML services: 1) user-based services, such as facial recognition and motion detection, and 2) utility-based services, including intrusion detection and intelligent optimization of data. In this paper, we consider one example service from each: facial emotion recognition and intrusion detection.

The user's generated data will be sent to the client database. Each service can have its own local database. The third-party service providers are the ones that the user has subscribed to, and they act as the aggregators in FL. An aggregator will forward a global model without accessing the original data. Local data will be consumed by two models for local training rounds. Then, these trained models are sent to the attack module, which acts as a penetration testing module that tests the resilience of the models against multiple types of attacks. The attack module also has access to the real database to verify the accuracy of the attacks. These attacks can be either existing or newly identified. The FL model should be resilient and continuously test for any privacy vulnerability against both new and existing attack types. The defence recommender server will securely send configurations that contain details on launching new penetration attacks. The attack module can perform them. The local defence recommender can evaluate the attack performance and provide recommendations by executing a defence mechanism to improve the model privacy based on user preference.



(a) Attacker poisons the target model by adding GAN data via malicious clients.

(b) Attacker gets target model predictions and trains shadow models and uses data from both to train attack model.

(c) Attacker records timestamps and gets FL-TIA with a change in membership probability.

Fig. 2. The proposed FL-TIA steps involve: (a) data generation phase, (b) attack model training phase, and (c) inference phase.

IV. Privacy Attacks for FL Resilience Testing

This section introduces the types of attacks we launch to test model resilience in the defence framework. For this, we use two types of attacks: 1) FL-TIA, which is a novel type of attack we propose, based on inferring membership inference variation over time, and 2) DLG attack, which is an existing model inversion attack that reconstructs the original dataset. The attacks were chosen specifically to illustrate the impact of two main privacy attacks: inference and data reconstruction. When considering the case of the possibility of such attacks on VR-based applications, a recent study in [20] shows consumer VR devices consist of many privacy and security vulnerabilities that can grant attackers access. Some of these consumer devices also do not have standard security features like multi-factor authentication, which makes them more vulnerable to attacks. These security and privacy gaps can easily be exploited by adversaries. Therefore, these attacks are designed to demonstrate and evaluate the scenarios of testing the FL model's resilience.

A. Attack Type 1: Membership Variation FL-TIA

The first attack we introduce is based on inferring on the property *time*. With FL, one can assume the ability of a third party to identify the period a data value used in training is mitigated by keeping data private. However, this proposed attack can reveal the interval when data is utilized in FL model training. For this, we use the change of membership inference over FL rounds. Rather than only using existing attack types, the primary objective of this novel attack against the defence mechanism is to test the effectiveness of the defence against new attacks and to demonstrate the defence can be effective against similar future inference attack types.

Similar to [3], [9], our work considers that the attacker can obtain model parameters. Then, they can observe how the model parameters change over time to infer the properties of the original dataset. The attacker may also possess a small sample of the original training data for the attacks. The attacker can obtain such data from compromised clients. In a real-world scenario, it may be possible to get data by attacking a few low-end IoT devices [21] or an IDS with a weaker security mechanism. Furthermore, data may not necessarily be the training data from real devices. Only a small representative sample similar to input data is sufficient for generating more data with GAN.

In a real-world scenario, the attacker can use multiple clients to conceal the attack. In addition, an attacker can be a man-in-the-middle eavesdropper who may track the model updates or push their adversarial model. If the attacker is a malicious aggregator, they can also follow and modify individual client nodes on each weight update, which can further refine the FL-TIA we propose. In the experiments, we consider the attacker has direct access to the target model updates. Fig. 2 provides an overview of the steps followed in our attack.

1) Attack Model With GAN Generated Data and Shadow Model: The first phases of the attack consist of poisoning the target FL model with attacker-owned data and training an attack model. Poisoning is the addition of GAN-generated data while training the target model. The attack model is an ML model that can classify the membership state of a certain data record, whether that data was included in the training dataset of the client FL model or not. For this, the attacker needs a dataset to train the attack model. This is obtained by the poisoned datasets since they know the membership state of the poisoned data. This process is further described as follows.

With a small representative dataset from FL clients, an attacker can generate any amount of attack input data D_{gen} using GAN models [3]. The labels for this GAN input data can be obtained by sending via the global model $f_t(.)$. A subset of D_{gen} dataset named $D_{gen}^{T_{in}}$ is used to poison and train the next t round of the target model $f_t(.)$. The poisoned model is then tested with another portion of the GAN dataset D_{gen}, which is named $D_{gen}^{T_{out}}$. Here, *out* signifies the membership state where this data is not included in the $f_t(.)$ training process. Using the combined predictions from $D_{gen}^{T_{in}}$ and $D_{gen}^{T_{out}}$, we then form the dataset D_{target}. To improve the overall accuracy of the attack, we combine the GAN dataset with the shadow model-based technique from [9]. These shadow models have the same architecture as the target ML model. We create m number of shadow models $\phi_i(.)$. Each $\phi_i(.)$, is trained with datasets $D_{gen}^{I_{in}}, D_{gen}^{I_{out}}$ which are subsets of D_{gen}. The results of training are aggregated as D_{shadow}.

The next objective of the attacker is to train attack model $\Omega^t(.)$ that can predict the membership states of a new data record at FL round t. As the training data for the attack model, we combine D_{target} and D_{shadow} and name as D_{attack}. Each

Algorithm 1 Create Attack Model for FL Model at Round t

Input: D_{gen} a GAN-generated dataset, $f_t(.)$: target FL model
Output: $\Omega^t(.)$: attack model for round t
function POISON()
  Let $[D_{gen}^{T_{in}}, D_{gen}^{T_{out}}] \subset D_{gen}$
  for next training round t **do**
    Train $f_t(D_{gen}^{T_{in}})$;
  end for
  Test $f_t(D_{gen}^{T_{out}})$
  Return D_{target}
end function
function SHADOW_TRAIN()
  $D_{shadow} = \{\}$
  for m shadow models, in each shadow model ϕ_i **do**
    Let $[D_{gen}^{I_{in}}, D_{gen}^{I_{out}}] \subset D_{gen}$
    Train $\phi_i(D_{gen}^{I_{in}})$ and then Test $\phi_i(D_{gen}^{i_{out}})$
    $D_{shadow} \leftarrow D_{shadow} \cup D_{shadow}^{i}$
  end for
  Return D_{shadow}
end function
function ATTACK_TRAIN()
$D_{attack} \leftarrow (D_{target} \cup D_{shadow})$
  for each input class j in D_{attack} **do**
    Let $D_{attack}^{j} \subset D_{attack}$
    Train $\Omega_j^t(y_{pr}^j, y_{gen}^j, s)$; $(y_{pr}^j, y_{gen}^j, s) \in D_{attack}^j$
  end for
  Return $\Omega^t \leftarrow \{\Omega_1^t, \Omega_2^t, ..., \Omega_c^t\}$
end function

record of D_{attack} is in the form (y_{pr}, y_{gen}, s), where y_{pr} denotes the predicted value after poisoning, y_{gen} is the prediction before poisoning and s as the membership state of the record. For each class label, we train a separate attack model and get the inference by considering all of their predictions. We summarise all mentioned steps in Algorithm 1.

2) Continuously Progressing Attack Model for Training Rounds: The attack model trained in FL training round t may not be as accurate in training round $t+1$ since new data may get added each round. To overcome this issue, we introduce the idea of the continuous progression of the attack model.

After getting the initial version of the attack model from the Algorithm 1, we follow a similar approach to get the attack models for each new round of FL target model training. We use Algorithm 2 to get the attack model Ω^{t+1} at next round $t+1$. Here, we get a representative sample D_{gen}^{t+1} for this new round $t+1$. Then, we combine D_{gen}^{t+1} to the target model training dataset D_{gen}^{t} used in the previous round. This combined dataset is used in the function POISON(), which will return D_{target}^{t+1}. This output added to the previously trained shadow model dataset D_{shadow}, represented as $(D_{target}^{t+1} \cup D_{shadow})$. It is then used to train the updated attack model Ω^{t+1} through the function ATTACK_TRAIN(). This approach takes less time to train the new attack model since we are using the existing shadow dataset and not training the shadow models again.

Algorithm 2 Progressing Attack Model for q Training Rounds

Input: In a new round $t+1$, D_{gen}^{t+1}; $D_{gen}^{t+1} \subset D_{gen}$
Output: $\{\Omega^1, \Omega^2, ..., \Omega^q\}$: A set of attack models for q training rounds
Let $A = \{\}$
for q rounds, in each training round **do**
  POISON() $\leftarrow$ *Input*: $D_{gen}^{t+1} \cup D_{gen}^{t}$; *Return*: D_{target}^{t+1}
  $D_{attack}^{t+1} \leftarrow (D_{target}^{t+1} \cup D_{shadow})$
  ATTACK_TRAIN() $\leftarrow$ *Input*: D_{attack}^{t+1}; *Return*: Ω^{t+1}
  $A = A \cup \Omega^{t+1}$
end for
Return $A = \{\Omega^1, \Omega^2, ..., \Omega^q\}$

Following this for q training rounds, we get a set of attack models $\{\Omega^1, \Omega^2, \ldots, \Omega^q\}$ for each round.

3) FL-TIA Using the Trained Attack Model: The updated attack models in each training round are used to infer the time interval when a new data element is added to the FL model training process. The malicious entity can maintain a timestamp for each FL model training round.

In between two training rounds t and $t+k$, we have two attack models Ω^t and Ω^{t+k} respectively. For a given data record r, the membership state probability can be obtained from the attack model Ω^t as p_t and Ω^{t+k} as p_{t+k}. For a threshold probability of P_K, if $p_t < P_K$ and $p_{t+k} > P_K$, an attacker can identify that the data record r has been entered into the FL model training in between the two rounds t and $t+k$. With this approach, we present the concept of continuous training of attack models in Algorithm 2, where the attacker can update attack models over progressing training rounds.

B. Attack Type 2: Deep Leakage From Gradients

In this attack, the attacker aims to reconstruct the private dataset used to train a model. This is done via an adversarial model that attempts to match the original gradients of the victim model [10]. We selected this since it is a well-established attack that is often used for benchmarking privacy mechanisms. Furthermore, data reconstruction attacks can result in significant privacy impact directly on individual data points rather than properties emerging from them, making them more attractive for attackers. Therefore, being resilient against this attack is important for our defence mechanism.

Let $f(x; W)$ be an FL model. An attacker randomly initializes a set of gradients $\nabla\tilde{W}$, where the target model gradients are ∇W. The loss of L_g between random and target model gradients can be represented in the following equation:

$$L_g = ||\nabla\tilde{W} - \nabla W||_f^2 \tag{1}$$

With this, the attacker can update a random data $\tilde{x}$ that attempts to reconstruct the original data x through gradient descent for N iterations as:

$$\tilde{x} \leftarrow \tilde{x} - \alpha\nabla_{\tilde{x}} L_g \tag{2}$$

where α is the learning rate.

An update for DLG is presented as iDLG in [11], which further enhances the original attack by first determining the

ground truth labels and adding them as a part of inputs for faster convergence. The amount of data necessary to train FL models will be limited when considering a single target VR device at the edge. Therefore, these models may get overfitted to the owner's data. It is relatively easier to reconstruct smaller datasets from the gradients [10], [11]. Therefore, if an attacker obtains these models via eavesdropping or getting internal access to VR headsets, it is highly likely to recover original data from captured gradients. These devices may have poorer privacy protections because of resource constraints, or the communication channels may have insufficient security, which makes it relatively simpler for an attacker to conduct an attack.

V. Attack Mitigation Through Split-Gradient and Perturbation (SAP) Technique

One of the main reasons for privacy attacks is the FL models get overfitted to the data at a given time window. This leads to inference or model inversion attacks if the attacker captures model parameters, such as weights or gradients. Therefore, as we discussed in Section II, an initial approach to mitigating attacks would be to add a controllable noise to the original model parameters through a technique such as DP. However, DP causes degradation of model utility since noise can directly impact model accuracy. Therefore, we propose a lightweight mechanism named SAP to mitigate attacks by splitting the model over time.

A. Process of SAP Mechanism

Suppose the gradients of the local model is ∇W, where $\nabla W = \frac{\partial L}{\partial W}$, which is a change of the loss function overweights of the NN model. We split ∇W to two components $\nabla W_1^{<t>}$ and $\nabla W_2^{<t>}$ as follows:

$$\nabla W_1^{<t>} = 2\frac{\partial L^{<t>}}{\partial W} + i^{<t>} \tag{3}$$

$$\nabla W_2^{<t>} = -\frac{\partial L^{<t>}}{\partial W} - i^{<t>} \tag{4}$$

where $i^{<t>}$ is a randomly added noise value independent of the gradients. At the start of time at $t = t$, we apply Stochastic Gradient Descent (SGD) only for $\nabla W_1^{<t>}$ with learning rate α.

$$W^{<t>} = W^{<t>} - \alpha\left(2\frac{\partial L^{<t>}}{\partial W} + i^{<t>}\right) \tag{5}$$

When performing SGD to the next training round at $t = t+1$ onward, we append $\nabla W_2^{<t>}$ from the previous round:

$$\frac{\partial L^{<t+1>}}{\partial W} = 2\frac{\partial L^{<t+1>}}{\partial W} + i^{<t+1>} + \nabla W_2^{<t>} \tag{6}$$

$$= 2\frac{\partial L^{<t+1>}}{\partial W} + i^{<t+1>} - \frac{\partial L^{<t>}}{\partial W} - i^{<t>} \tag{7}$$

When applying SGD at $t = t + 1$, we get,

$$W^{<t+1>} = W^{<t>} - \alpha\left(2\frac{\partial L^{<t+1>}}{\partial W} + i^{<t+1>} - \frac{\partial L^{<t>}}{\partial W} - i^{<t>}\right) \tag{8}$$

Applying (3) for $W^{<t>}$ in above equation gives,

$$W^{<t+1>} = W^{<t>} - \alpha\left(2\frac{\partial L^{<t>}}{\partial W} + i^{<t>}\right) - \alpha\left(2\frac{\partial L^{<t+1>}}{\partial W} + i^{<t+1>} - \frac{\partial L^{<t>}}{\partial W} - i^{<t>}\right) \tag{9}$$

Further simplifying (9) gives,

$$W^{<t+1>} = W^{<t>} - \alpha\left(\frac{\partial L^{<t>}}{\partial W}\right) - \alpha\left(2\frac{\partial L^{<t+1>}}{\partial W} + i^{<t+1>}\right) \tag{10}$$

which provides the result

$$W^{<t+1>} = W'^{<t>} - \alpha\left(2\frac{\partial L^{<t+1>}}{\partial W} + i^{<t+1>}\right) \tag{11}$$

where $W'^{<t>}$ is the desired weight at round t without adding the noise $i^{<t>}$. Therefore, we see the noise parameter added in the previous round t gets cancelled out over the next iteration; meanwhile, a new noise is added at round $t+1$. An overview of the mechanism is presented in Fig. 3.

Suppose the noise is not added in consecutive client rounds but between two random rounds t and $t + k$. In such a case, the noise will still be cancelling out after $t + k$ iteration. This can represented by extending Equation (10).

$$W^{<t+k>} = W^{<t>} - \alpha\left(\frac{\partial L^{<t>}}{\partial W} + \sum_{j=t+1}^{t+k-1}\frac{\partial L^{<j>}}{\partial W}\right) - \alpha\left(2\frac{\partial L^{<t+k>}}{\partial W} + i^{<t+k>}\right) \tag{12}$$

However, in this approach, the noise initially added in iteration t may have a relatively higher impact over the gradients of the next $k-1$ local iterations as noise does not get cancelled at consecutive rounds. Thus, noise can result in deviations from the original gradient descent without noise. Therefore, we use the application of noise masks at consecutive rounds in our implementations with minimum impact from the noise.

When considering global aggregation of models, for a local client l, the gradients that are sent to the server for aggregation after client round t can be obtained from Equation (8) as $d_l = \frac{\partial \tilde{L}_l^{<t>}}{\partial W} + \tilde{i}_l^{<t>}$, where $\frac{\partial \tilde{L}_l^{<t>}}{\partial W} = 2\frac{\partial L_l^{<t>}}{\partial W} - \frac{\partial L_l^{<t>}}{\partial W}$ and $\tilde{i}_l^{<t>} = i_l^{<t>} - i_l^{<t>}$. After aggregating the gradients by the server, the global gradient d_g sent back to clients is:

$$d_g = \sum_{l=1}^{n}\frac{\partial \tilde{L}_l^{<t>}}{\partial W} + \sum_{l=1}^{n}\tilde{i}_l^{<t>} \tag{13}$$

We use zero-mean local noise, where the accumulated noise component $\sum_{l=1}^{n}\tilde{i}_l^{<t>}$ averages zero according to the sum of random independent variables and the central limit theorem and results in a standard deviation of $\sigma\sqrt{\frac{2}{K}}$ for K number of clients when the local noise has σ standard deviation. Due to this cancelling noise, individual local model gradients are not directly readable by the aggregator, preserving the local model update's privacy. Thus, with a sufficient amount of clients, the

Algorithm 3 Implementing Privacy Recommendations

1: **Input:** Attack function $\psi_a()$, model training function with SAP enabled $\gamma()$, FL model M, threshold success rate θ_t
2: **Output:** FL model with privacy recommendation $\bar{M}$
3: Get $\psi_a(M) \rightarrow \theta_a$
4: **for** each training round i **do**
5: Let $\bar{M} = \gamma(M)$
6: $\psi_a(\bar{M}) \rightarrow \theta_a$
7: **if** $\theta_a \leq \theta_t$ **then Return** $\bar{M}$
8: **end if**
9: **end for**

SAP mechanism can provide a gradient accumulation like a normal aggregation process while maintaining privacy.

B. Implementing Privacy Recommendations

The attack module can perform an attack function $\psi_a()$ over the model M without privacy and obtain a metric θ_a on attack success. This metric is forwarded to the recommender. The recommender can have a predefined threshold level θ_t for attack success information metric, which can either be provided by the consumer, according to their preference of privacy setting, or by the service provider, meanwhile securely providing attack configurations. Then, the recommender can run the training rounds with SAP-enabled FL model training function as $\gamma(M)$, and in each round, they can check if $\theta_a \leq \theta_t$. Algorithm 3 provides the summary of the implementation.

The module can then successfully recommend the trained model to be sent to a third-party server. The IoT device can either continue training the model after this instance or can directly send this model to the aggregator.

In a case where the threshold is not reached, several changes can be made. First, the hyperparameters can be tuned, and the noise threshold levels can be adjusted to reach the threshold. If that does not work, the model should not be passed for the aggregation process since it can potentially leak the privacy of the clients. Therefore, the recommender can block sending the model to a third party. The client can wait until sufficient data is obtained to train it with a more generalised dataset. Clients can also wait for the next global aggregation round(s) until an updated model M is received from the server. Then, they can increase their model generalisability by aggregating with the global model with reduced overfitting, which will be helpful in reaching the threshold limits faster.

C. Comparison With Differential Privacy-Based Noise

The DP-based deep learning was first introduced in [22] with Differentially Private Stochastic Gradient Descent (DP-SGD), where random noise is added at each training step. In DP-SGD, the gradients are clipped with clipping threshold C:

$$\nabla\tilde{W}(x_i) \leftarrow \nabla W(x_i)/max\left(1, \frac{||\nabla W(x_i)||_2}{C}\right) \tag{14}$$

where x_i is an element in the dataset of N total elements with group size L. Then, Gaussian noise $i(0, \sigma^2 C^2 \mathbf{I})$ is added for each gradient:

$$\nabla\tilde{W} \leftarrow \sum_i \left(\nabla\tilde{W}(x_i) + i\right) \tag{15}$$

Therefore, unlike our method, the random addition of noise will get compounded over the iterations as shown by Equation (15). This results in reduced model utility over time since the accuracy of the model tends to degrade with compounding noise in the gradients.

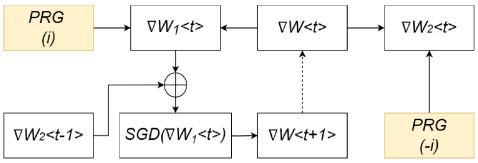


Fig. 3. SAP process of splitting the local model gradients and noise addition with Pseudo-Random Generator (PRG).

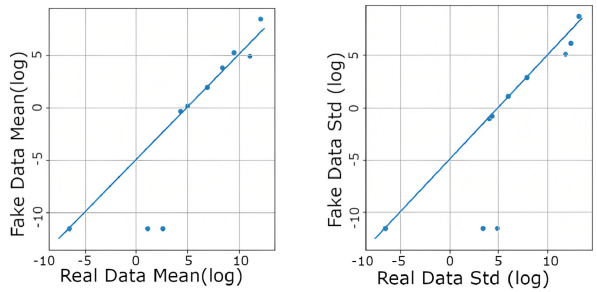
VI. Experiments

In this section, we present the experiments and simulations we performed on the two components of the framework. First, we discuss the novel inference attack type FL-TIA, which is launched by the attack module for resilience testing. Next, we discuss how our proposed defence mechanism can protect privacy, which is done at the privacy recommender module.

A. Launching FL-TIA Against FL Models

This section discusses the experimental procedure used to evaluate the FL-TIA on a simulation of IDS-based FL models in VR devices. For our experiments, we used the dataset NSL-KDD, often used for IDS simulations, which consists of 125,973 data records providing details on network traffic based on various network attack scenarios. From the dataset, we pick the two most frequent types of traffic for our experiments: normal and Denial of Service (DoS) attacks. All the experiments were implemented using Keras and Tensorflow Federated (TFF) framework. We set the configurations in TFF for both clients and the aggregator to train the models automatically with SGD optimization. For the dataset, we used 70% training vs. 30% test set. The training set was equally split among the clients. We run the experiments using a computing instance with an Intel Xeon 2.20 GHz CPU, 26GB RAM, and an NVIDIA Tesla T4 GPU. We considered the target model a sequential NN with a hidden layer of 512 dense units followed by a dropout layer with a 0.2 dropout rate. The attack was simulated for multiple clients, where we trained the initial global model with 50 clients using the original data. Initially, we ran 20 rounds to make the FL target model converge to a stable accuracy.

As the next step, we trained a GAN to generate representative data for the attacker. We used 1,000 samples from the original dataset to train the generator and discriminator models. This resembles an attacker getting a small percentage of data as a representative sample of the original dataset. The generated dataset from GAN is unlabelled. Therefore, we used the trained FL target model to get labels by sending



(a) Mean (b) Standard deviation

Fig. 4. GAN model training metrics for NSL-KDD dataset.

TABLE I
METRICS FOR FL-TIA ATTACK MODEL AT ROUND 21

Metric	Only shadow models	Only GAN-based poisoning	**Both GAN and shadow models**
Test accuracy	0.6617±0.012	0.6897±0.021	**0.8183±0.016**
Precision	0.5673±0.017	0.5716±0.016	**0.6529±0.013**
Recall	0.6529±0.017	0.7085±0.011	**0.9200±0.027**
F1	0.5931±0.013	0.6267±0.014	**0.7635±0.013**

the generated data through the target model. The generated results show a high similarity with reduced error over GAN training when compared with original training sets, as shown in Fig. 4. For the NSL-KDD dataset, we compare the mean and standard deviation values of the numerical attributes present in the GAN-generated sample data with the mean values of the original dataset in the log scale. The results show the values lie close to the actual mean, with a similar standard deviation. The Generated dataset, therefore, can be considered a representative of the original data used in FL.

To create the attack model, we poison the target models using the generated labelled data with 20,000 records for the NSL-KDD. Here, we evenly distribute GAN-generated data for model training among the clients. We also used 10,000 generated data records to train five shadow models. The NN architecture of the shadow models is set similarly to the target model. We trained the shadow models for 20 rounds. The shadow models produce 50,000 shadow training outputs, which we use to train our attack models further. After that, we train the attack model with the data from the poisoned target model and the shadow models. It is a Multilayer Perceptron (MLP) classifier with two hidden layers of 64 and 16 units.

1) Use of Combined GAN and Shadow Model for FL-TIA: The proposed FL-TIA uses both GAN and shadow models to launch the attack. This is done since, from the attacker's perspective, real data is not available to the attacker. They may possess a small representative of data where they enrich the dataset through GAN. By training shadow models, the attacker can get further information on how the FL model gets trained. We compared both approaches, and our results in Table I show that combining both methods gives the highest attack accuracy.

2) Maintaining Accuracy of Attack Models Over FL Model Training Rounds: When using the same attack model trained for a previous round for predicting membership states, we observe that the accuracy of the attack model is gradually reduced. None of the previous works of the current state-of-the-art, such as [3], [9], considered the factor of loss of accuracy over iterations. In our attack, we introduce the approach of continuous progress of the attack model to maintain its accuracy. For this, we follow the approach described in Algorithm 2, where we train a new attack model in each round by continuously poisoning the target model.

Algorithm 4 Evaluating FL-TIA Through Round Prediction

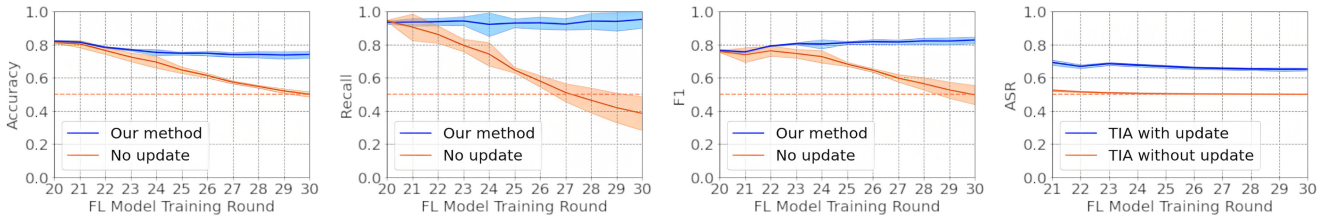
1: **Input:** Representative dataset $D^t_{gen} \leftarrow (y^t_{pr}, y^t_{gen})$, attack model Ω^t for round t, attack model Ω^{t+k} for round $t+k$
2: **Output:** The Attack Success Rate ASR
3: $s = 0 \forall$ elements in D^t_{gen}
4: Get $p_0 = \frac{\sum_{i=1}^{|D^t_{gen}|}(\Omega^t((y^t_{pr_i}, y^t_{gen_i}))=0)}{|D^t_{gen}|}$
5: At round $t+k$; POISON()$\leftarrow$ *Input*: D^t_{gen}; *Return*: D^{t+k}_{gen} where $s = 1 \forall$ elements in D_{v1}
6: Get $p_1 = \frac{\sum_{i=1}^{|D^{t+k}_{gen}|}(\Omega^{t+k}((y^{t+k}_{pr_i}, y^{t+k}_{gen_i}))=1)}{|D^{t+k}_{gen}|}$
7: **Return** $ASR = \frac{p_1+p_0}{2}$

To compare our approach with the existing works, we continue to train attack models for a further 10 FL rounds with Algorithm 2 using 20,000 GAN data records of NSL-KDD. We also evaluate the metrics over the FL rounds with the attack model without further training, as presented in related work [3]. We tested the accuracy of the attack models with 5,000 poisoned data records for NSL-KDD, having a similar distribution of membership states. From the results in Fig. 5(a), we observe that the attack model continuously updated with our approach maintains the initial accuracy, while the accuracy of the attack model without any updates is reducing over the rounds. Similarly, recall and F1 metrics in Fig. 5 illustrate that our attack is capable of maintaining the scores at higher values throughout the training rounds. Therefore, our attack can consistently maintain the accuracy of the attack model.

3) Time Inference Attack via Round Prediction: To evaluate the FL-TIA success rate over multiple rounds, we designed an experiment summarised in the Algorithm 4. Here, for each round t, we get a sample GAN dataset D^t_{gen} where initial membership state $s = 0$ for all elements in D^t_{gen}. We send this data through the attack model, Ω^t, for round t. The membership probability p_s for a round u in a dataset D^u_{gen} can be obtained as:

$$p_s = \frac{\sum_{i=1}^{|D^u_{gen}|}\left(\Omega^u(y^u_{pr_i}, y^u_{gen_i}) = s\right)}{|D^u_{gen}|} \tag{16}$$

where s is the membership state of either *in* or *out*. Using this equation, we obtain the membership probability p_0 of round t from Ω^t, with a GAN dataset D^t_{gen} where initially $s = 0$ in all elements in D^t_{gen}. Then, we poison the FL model with the same dataset over another round $t+k$. Here the dataset is now called D^{t+k}_{gen} and membership state $s = 1$ for all elements in D^{t+k}_{gen}. Next, the dataset is tested with the attack model Ω^{t+k}



(a) Accuracy. (b) Recall. (c) F1. (d) FL-TIA success rate.

Fig. 5. The accuracy, recall, F1 of attack models and FL-TIA success rate with and without continuous updates for NSL-KDD including 95% confidence intervals for each FL round.

specialized for round $t+k$. After that, membership probabilities p_1 predicted by Ω^{t+k} is obtained.

In the experiment, we train attack models for each new round from round 21 up to round 30. For each round, we use a sample GAN dataset with 5,000 records for NSL-KDD as testing data across two rounds to evaluate the FL-TIA. We get the round predictions for the interval between each new attack model and the attack model in FL round 20, based on Algorithm 4. From this, we obtain the Attack Success Rate (ASR) for both with and without the continuously updated attack models. Here, we defined ASR as an average value for a success rate of the correct membership predictions between the two rounds. The results are shown in Fig. 5(d). They show FL-TIA has a higher success rate with our approach of using continuously updating attack models.

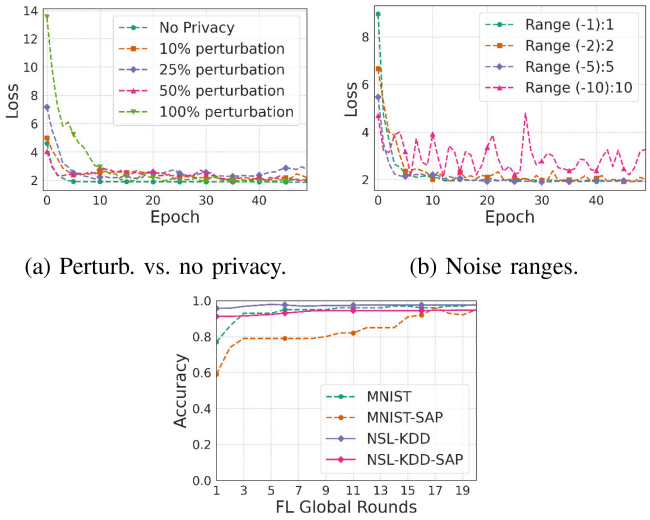
4) Relationship Between FL-TIA via Round Prediction and Membership Inference: FL-TIA through round prediction depends on the accuracy of the attack models of membership inference since our attack is based on identifying the membership change for a data record between two rounds. We can observe that the ASR of round predictions in Fig. 5(d) has an accuracy of around 0.6 with our approach. However, for both cases, the attack models' accuracy is relatively higher than the round prediction. This may occur because, in a round prediction attack, we evaluate data using two attack models where ASR depends on the accuracy of the two models. It would be approximately equal to the probability:

$$ASR \approx p_t \times p_{t+k} \tag{17}$$

where p_t and p_{t+k} are the accuracies of the attack models at rounds t and $t+k$ respectively.

B. SAP Defence Mechanism Against the Attacks

In defending against privacy attacks, we use our SAP defence technique described in Section V, where gradients are split into two components and added with random noise for gradient value corresponding to each parameter in the model. The noise gets cancelled over the next rounds; therefore, the impact on the overall model utility from the defence is low. We test this by implementing the mechanism on a LeNet-5 model, where we use the Facial Expression Recognition (FER-2013) dataset to train the model. The dataset resembles a scenario of VR-based facial data where the FL model is used to identify facial expressions in real time. The FER-2013 dataset consists of 7 types of expressions with 28,709 training



(a) Perturb. vs. no privacy. (b) Noise ranges.

(c) FL aggregation rounds.

Fig. 6. Convergence of NN model with (a) adding noise to the NN layer with changing probability, (b) varying noise ranges adding with 100% probability and (c) variation of the accuracy of FL global models over 20 FL rounds.

and 3,589 testing data in 48x48 pixel grayscale images. We used 50 random images covering all categories and trained the LeNet-5 model for 50 iterations.

1) Impact on FL Model: First, we analyse the impact of the defence over the model convergence to identify if there are any utility trade-offs of the mechanism. For the SAP mechanism noise mask, we selected a uniform noise function that is in the float range from –5 to 5. The loss values for different levels of perturbations are presented in Fig. 6(a).

From the results, we observe the model converges since the loss is reduced over all the levels of perturbations. The addition of perturbations at higher levels results in higher losses at the beginning, but the model eventually reaches stable convergence over the local iterations.

We also tested the effect of different noise ranges on the model convergence. We selected four different ranges shown in Fig. 6(b). From the results, we see that when the noise levels increase, more disturbances can be seen during the model convergences. However, since the noise gets cancelled out in the following iterations, they reach convergence. Therefore, selecting intermediate noise levels can provide better privacy protection than lower ranges, which also provides a similar convergence and model utility. The benefit of having higher

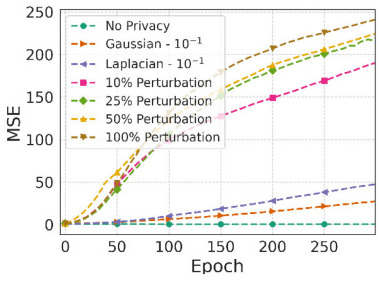


Fig. 7. Variation of MSE of the iDLG attack.

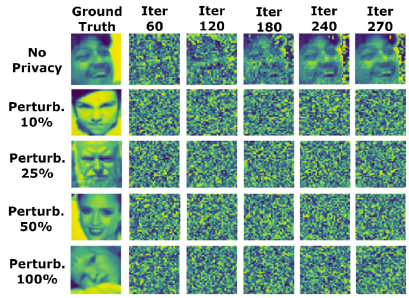


Fig. 8. Data reconstruction results without privacy implementation vs. different levels of privacy using our SAP mechanism.

perturbations is enhanced privacy, which we evaluate in the next experiment to analyse the resilience of the mechanism against the deep leakage attack.

Furthermore, we evaluated how two different NN models globally converge with the mechanism. Here, in addition to NSL-KDD, we also used the MNIST dataset, which is an image dataset of 60,000 records of handwritten digits. For experiments, we used two NN architectures for two datasets: the sequential model used in Section VI-A for NSL-KDD and a Convolutional Neural Network (CNN) architecture with 2 convolutional layers followed by 2 dropout and sequential layers. We used 10 sample clients, where each client has 1,000 data records. The experiment was run for 20 FL aggregation rounds without privacy and with the SAP mechanism. Results are shown in Fig. 6(c). From the results, it can be observed that different model architectures can have varying impacts on the aggregation with the SAP mechanism at the beginning of the global aggregation rounds. Initially, MNIST models seem to have relatively lower accuracy in the initial rounds compared to their corresponding no-privacy implementation. Compared with that, NSL-KDD has an initial accuracy that is similar to the original FL models without privacy. As discussed in Section V, the SAP mechanism has a similar aggregation accuracy to the original update since the impact from noise on accuracy is mitigated due to noise averaging to zero and resulting near similar gradient updates to the original gradients.

2) Defence Against Deep Leakage Attack: In this experiment, we test the performance of SAP defence against an existing attack scenario on client data. For this, we use the improved version of the Deep Leakage iDLG attack [11], where the attack module launches this attack on the client model. The main difference between DLG and iDLG is that iDLG has better attack convergence capability, as discussed in Section IV. Therefore, iDLG is a more challenging attack than DLG, yet their main procedure of inversion of gradients remains similar. Thus, if the defence mechanism provides protection against iDLG, it will inherently protect against the DLG. Following this, the attack module attempts to reconstruct the original data used to train the model by inverting the gradients. To perform the attack, we train the LeNet-5 for 5 iterations with small amounts of data, with 5 face images, since small batches are more vulnerable to attack. The trained model is then attempted with the iDLG, without and with our SAP mechanism. The error incurred in reconstructing the original dataset is presented in Fig. 7.

Here, we observe that without the privacy mechanism, the model is highly vulnerable to attack, and the images can successfully be reconstructed via iDLG. However, with our defence, the Mean Squared Error (MSE) threshold metric increases over the iterations for all percentages of the perturbations added via the mechanism. This results in increasing difficulty for the attacker in reconstructing the original data. Therefore, the method can provide resilience against iDLG attacks, even at low quantities of data with more vulnerability. Furthermore, we compared the MSE of the attack with Gaussian and Laplacian noise, which are widely used in DP [10]. Similar to the original work of DLG in [10], we set the threshold variance at 10^{-1} and mean 0 for both noise ranges since the iDLG attack also has the same procedure of DLG for calculating the loss between the gradients. Fig. 7 shows that, compared with the privacy enhancements from Gaussian and Laplacian noise, our mechanism provides a significantly high reconstruction error, thus providing better privacy for the models. Fig. 8 shows how the reconstruction of sample data is observed at different iterations of the attack.

Results show our method does not produce any identifiable features for all of the perturbation levels.

3) Defence Against FL-TIA Membership Variation Detection: The membership variation detection FL-TIA attack is the other type of attack we introduced in Section IV, where the attacker intends to reveal time-related information in the local IDS model in the consumer's VR headset. To verify if our proposed SAP mechanism can defend against the FL-TIA with membership variation detection, we apply the defence to the same target NN model used for FL-TIA.

When considering the minimum baseline threshold attack accuracy required to provide defence recommendations, we select 50% accuracy for inference attacks similar to [23], [24]. Therefore, for a particular client, an attacker should no longer guess the presence of a time-related attribute more than the random guessing accuracy once the defence is implemented. We follow Algorithm 3 to obtain this threshold by varying the local model SAP parameters like perturbation ranges and perturbed NN layers. To compare SAP with existing techniques, we also launched the widely adopted defence DP-SGD [22] against the attack FL-TIA. For DP, this value is obtained by varying the privacy budget (ϵ) parameter.

For the SAP mechanism with 1 layer perturbed, we obtain a local model accuracy of 95.16%, which is very close to the

TABLE II
DEFENCE METHOD COMPARED WITH DP AGAINST MEMBERSHIP VARIATION DETECTION

DP Status	FL Accuracy	ASR
No Privacy	0.9578±0.005	0.6925±0.015
DP $\epsilon = 4.0$	0.9515±0.004	0.6538±0.023
DP $\epsilon = 2.0$	0.9449±0.002	0.5869±0.016
DP $\epsilon = 1.0$	0.9402±0.002	0.5708±0.003
DP $\epsilon = 0.5$	0.8605±0.001	0.4999±0.001
Our Approach	0.9516±0.002	0.5007±0.042

TABLE III
TIME CONSUMPTION FOR RUNNING LOCAL EPOCHS

Num. Epochs	No Privacy (s)	DP (s)	SAP (s)
100	0.4129±0.062	0.9578±0.005	0.5116±0.010
1,000	2.8802±0.509	8.0977±1.008	3.594±0.179
10,000	36.5243±1.539	94.4242±1.656	37.8149±0.621

original model's accuracy of 95.78% without privacy. Then, we compare the success rate of the membership attack ASR with the mechanism, which we obtain as 50.07%. The ASR without privacy is recorded as 69.25%, and therefore, we see a significant improvement in privacy against the attack by lowering the ASR. In our defence, the ASR is very similar to the random guessing probability. For DP, We use different levels of ϵ where lower values provide a higher privacy guarantee with more noise added to the model. The results are summarised in Table II. From the results, we see the DP-SGD can reduce ASR and improve privacy, where at the privacy level of $\epsilon = 0.5$, it provides a lower ASR similar to our proposal. However, the utility of the model has also significantly reduced at this level, where the FL target model accuracy has dropped to 86.05%. Therefore, the comparison shows our SAP method can successfully defend against the attacks; meanwhile, it is capable of maintaining the model utility similar to the original performance over the iterations.

4) Time Consumption for Defence: To experimentally evaluate the time taken for the defence mechanism, we performed experiments for the NSL-KDD dataset sample of 10,000 training data in a model running 1,000 local iterations. For the DP, we set privacy level $\epsilon = 1$. For SAP, we set a 100% perturbation percentage applied to layer 1 of the model, following a similar configuration to the previous experiment. The same model configurations for the NSL-KDD dataset specified in Section VI-B1 were used in the experiments. The average times taken are shown in Table III.

From the results, it can be observed that the addition of our SAP mechanism with only a small perturbation mask does not result in any significant impact on the overall time taken for a local client, especially when increasing the number of iterations. However, it is clear that compared to our approach, DP can result in significantly high time due to its inner iterative implementation of the addition of noise and clipping over all the model parameters, thus resulting in higher computation costs. Furthermore, the mechanism does not alter the existing size of the client's local model. It only adds the noise mask to the existing model gradients. Therefore, when sharing the local model with third parties, it transfers the same model size as a no-privacy model, resulting in a similar latency.

VII. DISCUSSION

In this section, we discuss different aspects of the proposed novel FL-TIA attack and improvements to be made in the defence mechanism. Furthermore, we highlight the significance of our work compared with related works.

A. Factors Affecting the Defence and Trade-Offs

The main factor that can mitigate the attack is the noise addition by the defence mechanism in each round. Other factors include the nature of the noise since random uniform noise added in the process does not concentrate towards a mean value like Gaussian or Laplacian, which makes it difficult to eliminate the noise by the attacker unless it is removed explicitly by the original source at a later round.

Suppose an attacker eavesdrops on consecutive local models sent to the server. In that case, they may not be able to obtain the original model update by summation of the two models, even if the client runs only one local client iteration and sends the model right away since there will still be a noise mask for the next round implemented on it. By analysing multiple consecutive models, the attacker may try to estimate noise ranges or coefficients of the decomposed gradients. The defence technique can, therefore, vary these parameters to add an extra layer of protection. However, the higher noise ranges or varying coefficients may affect overall local and global model convergence, which may also need to be considered.

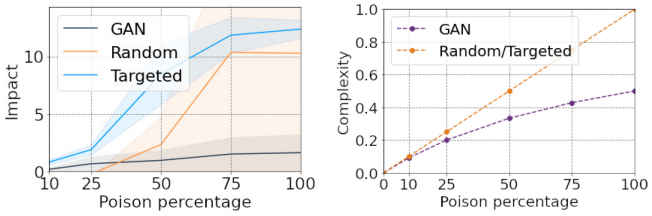
B. Impact of Poisoning on FL Model Accuracy

1) Aim of the GAN-Based Poisoning Phase: In our proposed FL-TIA, we perform poisoning of the FL models using a GAN-generated dataset. Unlike many poisoning attacks that distort the model classification, GAN-based poisoning is only done to get training data for the attack model. Therefore, the main intention of this poisoning is to provide an effective representation of the original dataset, which the attacker should be able to train the attack model with the GAN-based data.

2) Comparing GAN-Based Poisoning Impact and Complexity With Other Poisoning Types: One of the ways to verify if the GAN-generated dataset successfully represents the original data is by checking the impact on a local client's model accuracy before and after poisoning. If GAN-based data is a good fit, it should not change the accuracy significantly. To calculate the impact, we use equation (18) as follows:

$$Impact = \frac{Error(F_p) - Error(F)}{Error(F)} \tag{18}$$

where $Error(F_p) = 1 - F1$. For further comparison, we used three types of poisoning: 1) adding new GAN-generated data, 2) random swapping of labels within the original dataset, and 3) targeted poisoning, where labels are poisoned with only one specific targeted class label. For this, we use 1000 original data samples of NSL-KDD. For GAN-based poisoning, we gradually added new GAN data examples up to another 1000



(a) Impact of poisoning. (b) Complexity of poisoning.

Fig. 9. (a) Impact and (b) complexity of GAN-based poisoning compared with random and targeted poisoning for NSL-KDD.

values, totalling the dataset to 2000 with the original data and denoted as 100% poisoning. For other types of poisoning, we replaced the labels of the existing 1000 samples up to 100%. Fig. 9(a) shows the results obtained for the impact of the poisoning of the dataset. Relative to the other poisoning attacks, GAN-based poisoning has a lesser impact on the model utility, which confirms that the GAN dataset is a better representation of the original data. It also shows that such a GAN-based attack is relatively difficult to determine from observing model metric variation when compared with other poisoning approaches. Thus, the defence mechanism may be more feasible than attempting to detect GAN-based poisoning.

To measure how complex the poisoning attack is, we used the ratio of the poisoned data with the total dataset size. This is a measure with the dataset to evaluate how feasible the attack is to be performed in practice, as represented in equation (19):

$$Complexity = \frac{|D_p|}{|D + D_p|} \tag{19}$$

where D_p is the poisoned dataset amount, and D is the amount of benign data in the dataset. The more poisoned data in the total dataset, the more complex the attack since it will practically be increasingly difficult to poison the dataset with minimum noticeable effort. Furthermore, it will be more difficult to modify existing samples of data in private clients rather than introduce new data into the dataset. Thus, GAN-based poisoning is less complex than the two other approaches. The results for complexity are presented in Fig. 9(b).

C. Limitations and Potential Enhancements on Defence

The proposed mechanism does not provide a privacy bound compared with DP. However, as we have shown in Section VI, tighter privacy bounds can lead to utility loss. Similarly, DP can be more successful on large-scale clients who have large datasets, which may be unlikely at a single IoT consumer level. Therefore, as a potential enhancement to our existing solution, we can make a hybrid approach by adding quantifiable DP at intermediate client levels of a hierarchical FL architecture; meanwhile, the SAP mechanism can be used at individual consumer levels.

D. Comparison With Related Work

We compared our research with multiple associated works that are in the FL domain. A summary of the comparison is presented in Table IV. It shows that none of the related works includes all the aspects we considered. Especially when considering IoT, no related works provide a comprehensive framework for privacy recommendations for models via the adversarial ML approach we presented.

TABLE IV
SUMMARY CONTRIBUTION OF OUR WORK

Characteristics	Ref [25]	Ref [16]	Ref [18]	Ref [26]	Ref [27]	**Our work**
Privacy solutions for consumer IoT	M	L	L	M	L	**H**
Novel framework focused on privacy enhancement for FL models	L	H	H	M	H	**H**
Introducing new attack type on FL	L	H	L	L	L	**H**
Continuous attack accuracy over time for attack without defence	L	L	L	L	L	**H**
Design of new defence mechanism	H	M	M	M	H	**H**
Evaluating resilience of FL models against new and existing attacks	M	L	L	L	M	**H**
Maintaining model utility by FL defence	M	L	L	M	M	**H**
Achieve high defence privacy levels for multiple attack types	M	M	L	M	L	**H**

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

VIII. CONCLUSION

In this research, we evaluated the privacy of consumer IoT devices with FL-based services and introduced privacy recommendations based on the resilience of FL models against attacks. We presented a privacy-enhancing framework for FL-based services to perform privacy updates locally in the consumer device. This framework can perform privacy tests by launching attacks against FL local models from different services. For this, we performed an existing iDLG attack to reconstruct data, and we also performed a new type of attack, FL-TIA, based on time. The results show that plain FL models are highly vulnerable to attacks and can leak unintended user information. Therefore, we further introduced a novel privacy mechanism named SAP based on splitting gradients and adding perturbation masks over time. The experiments show our method can significantly outperform existing privacy-preserving techniques of random perturbations and DP without degrading FL model utility over time. In the future, we plan to extend the framework to P2P systems, where consumer devices can fully decouple from centralised FL-based services.

REFERENCES

[1] H. Hu, Z. Salcic, L. Sun, G. Dobbie, and X. Zhang, "Source inference attacks in federated learning," in *Proc. IEEE Int. Conf. Data Min. (ICDM)*, 2021, pp. 1102–1107.
[2] J. Gao et al., "Secure aggregation is insecure: Category inference attack on federated learning," *IEEE Trans. Depend. Secure Comput.*, vol. 20, no. 1, pp. 147–160, Jan./Feb. 2023.
[3] J. Zhang, J. Zhang, J. Chen, and S. Yu, "GAN enhanced membership inference: A passive local attack in federated learning," in *Proc. IEEE Int. Conf. Commun.*, 2020, pp. 1–6.
[4] L. Li, Y. Fan, M. Tse, and K.-Y. Lin, "A review of applications in federated learning," *Comput. Ind. Eng.*, vol. 149, Nov. 2020, Art. no. 106854.
[5] L. Yang, B. Tan, V. W. Zheng, K. Chen, and Q. Yang, "Federated recommendation systems," *Federated Learning* (Privacy Incentive). Cham, Switzerland: Springer, 2020.
[6] B. Tan, B. Liu, V. Zheng, and Q. Yang, "A federated recommender system for online services," in *Proc. 14th ACM Conf. Recommen. Syst.*, 2020, pp. 579–581.

[7] X. Yin, Y. Zhu, and J. Hu, "A comprehensive survey of privacy-preserving federated learning: A taxonomy, review, and future directions," *ACM Comput. Surveys*, vol. 54, no. 6, p. 131, 2021.
[8] Z. Wang, Y. Huang, M. Song, L. Wu, F. Xue, and K. Ren, "Poisoning-assisted property inference attack against federated learning," *IEEE Trans. Depend. Secure Comput.*, vol. 20, no. 4, pp. 3328–3340, Jul./Aug. 2023.
[9] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *Proc. IEEE Symp. Security Privacy (SP)*, 2017, pp. 3–18.
[10] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 14747–14756.
[11] B. Zhao, K. R. Mopuri, and H. Bilen, "iDLG: Improved deep leakage from gradients," 2020, *arXiv:2001.02610*.
[12] M. Kim, O. Günlü, and R. F. Schaefer, "Federated learning with local differential privacy: Trade-offs between privacy, utility, and communication," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, 2021, pp. 2650–2654.
[13] E. Sotthiwat, L. Zhen, Z. Li, and C. Zhang, "Partially encrypted multi-party computation for federated learning," in *Proc. IEEE/ACM 21st Int. Symp. Clust. Cloud Internet Comput. (CCGrid)*, 2021, pp. 828–835.
[14] H. Fang and Q. Qian, "Privacy preserving machine learning with homomorphic encryption and federated learning," *Future Internet*, vol. 13, no. 4, p. 94, 2021.
[15] G. Peralta, R. G. Cid-Fuentes, J. Bilbao, and P. M. Crespo, "Homomorphic encryption and network coding in IoT architectures: Advantages and future challenges," *Electronics*, vol. 8, no. 8, p. 827, 2019.
[16] C. Ma et al., "On safeguarding privacy and security in the framework of federated learning," *IEEE Netw.*, vol. 34, no. 4, pp. 242–248, Jul./Aug. 2020.
[17] Y. Liu, J. Peng, J. Kang, A. M. Iliyasu, D. Niyato, and A. A. Abd El-Latif, "A secure federated learning framework for 5G networks," *IEEE Wireless Commun.*, vol. 27, no. 4, pp. 24–31, Aug. 2020.
[18] L. Ouyang, F.-Y. Wang, Y. Tian, X. Jia, H. Qi, and G. Wang, "Artificial identification: A novel privacy framework for federated learning based on blockchain," *IEEE Trans. Comput. Soc. Syst.*, early access, Feb. 1, 2023, doi: 10.1109/TCSS.2022.3226861.
[19] Y. Liu, K.-F. Tsang, C. K. Wu, Y. Wei, H. Wang, and H. Zhu, "IEEE P2668-compliant multi-layer IoT-DDoS defense system using deep reinforcement learning," *IEEE Trans. Consum. Electron.*, vol. 69, no. 1, pp. 49–64, Feb. 2023.
[20] N. Noah, S. Shearer, and S. Das, "Security and privacy evaluation of popular augmented and virtual reality technologies," in *Proc. IEEE Int. Conf. Metrol. eXtended Real. Artif. Intell. Neural Eng. (IEEE MetroXRAINE 2022)*, 2022, pp. 1–6.
[21] F. Meneghello, M. Calore, D. Zucchetto, M. Polese, and A. Zanella, "IoT: Internet of Threats? a survey of practical security vulnerabilities in real IoT devices," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 8182–8201, Oct. 2019.
[22] M. Abadi et al., "Deep learning with differential privacy," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2016, pp. 308–318.
[23] L. Song, R. Shokri, and P. Mittal, "Membership inference attacks against adversarially robust deep learning models," in *Proc. IEEE Security Privacy Workshops (SPW)*, 2019, pp. 50–56.
[24] N. Carlini, S. Chien, M. Nasr, S. Song, A. Terzis, and F. Tramer, "Membership inference attacks from first principles," in *Proc. IEEE Symp. Security Privacy (SP)*, 2022, pp. 1897–1914.
[25] C. Zhang, S. Ekanut, L. Zhen, and Z. Li, "Augmented multi-party computation against gradient leakage in federated learning," *IEEE Trans. Big Data*, early access, Sep. 22, 2022, doi: 10.1109/TBDATA.2022.3208736.
[26] S. Truex et al., "A hybrid approach to privacy-preserving federated learning," in *Proc. 12th ACM Workshop Artif. Intell. Security*, 2019, pp. 1–11.
[27] J. Li, A. S. Rakin, X. Chen, Z. He, D. Fan, and C. Chakrabarti, "ResSFL: A resistance transfer framework for defending model inversion attack in split federated learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 10194–10202.

Chamara Sandeepa (Student Member, IEEE) is currently pursuing the Ph.D. degree with the School of Computer Science, University College Dublin, Ireland, and a Research Engineer for the EU H2020 SPATIAL Project.

Bartlomiej Siniarski (Member, IEEE) is currently a Postdoctoral Researcher and a Project Manager for the EU H2020 SPATIAL Project with University College Dublin.

Shen Wang (Senior Member, IEEE) is currently an Assistant Professor with the School of Computer Science, University College Dublin, Ireland.

Madhusanka Liyanage (Senior Member, IEEE) is an Assistant Professor/Ad Astra Fellow and the Director of Graduate Research with the School of Computer Science, University College Dublin, Ireland.