TU Delft

Delft University of Technology

Document Version
Final published version

Licence
CC BY

Citation (APA)
Ghavamipour, A. R., Turkmen, F., Wang, R., & Liang, K. (2023). Federated Synthetic Data Generation with Stronger Security Guarantees. In *SACMAT 2023 - Proceedings of the 28th ACM Symposium on Access Control Models and Technologies* (pp. 31-42). (Proceedings of ACM Symposium on Access Control Models and Technologies, SACMAT). Association for Computing Machinery (ACM). https://doi.org/10.1145/3589608.3593835

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Copyright
In case the licence states “Dutch Copyright Act (Article 25fa)”, this publication was made available Green Open Access via the TU Delft Institutional Repository pursuant to Dutch Copyright Act (Article 25fa, the Taverne amendment). This provision does not affect copyright ownership.
Unless copyright is transferred by contract or statute, it remains with the copyright holder.

Sharing and reuse
Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

Takedown policy
Please contact us and provide details if you believe this document breaches copyrights.
We will remove access to the work immediately and investigate your claim.

This work is downloaded from Delft University of Technology.

Federated Synthetic Data Generation with Stronger Security Guarantees

Ali Reza Ghavamipour
University of Groningen
The Netherlands
a.r.ghavamipour@rug.nl

Fatih Turkmen
University of Groningen
The Netherlands
f.turkmen@rug.nl

Rui Wang
Delft University of Technology
The Netherlands
r.wang-8@tudelft.nl

Kaitai Liang
Delft University of Technology
The Netherlands
kaitai.liang@tudelft.nl

ABSTRACT

Synthetic data generation plays a crucial role in many areas where data is scarce and privacy/confidentiality is a significant concern. Generative Adversarial Networks (GANs), arguably one of the most widely used data synthesis techniques, allow for the training of a model (i.e., generator) that can generate real-looking data by playing a min-max game with a discriminator model. When multiple organizations are reluctant to share their sensitive data, GANs models can be trained in a federated manner, commonly with the use of differential privacy (DP). In order to achieve a reasonable level of model utility, DP trades privacy exhibiting vulnerability to various attacks (e.g., membership inference attack). In this paper, we propose a hybrid solution, PP-FedGAN, to the asynchronous federated, privacy-preserving training of GANs models by combining the CKKS homomorphic encryption (HE) scheme with differential privacy. The addition of HE results in around 10 seconds of overhead on the client side per round and 115 seconds on the entire training procedure. We also analyze the security of PP-FedGAN under the honest-but-curious security model. Where stronger security guarantees are required, our proposal presents a better alternative to solutions that only employ DP.

CCS CONCEPTS

• **Security and privacy → Privacy-preserving protocols**; • **Computing methodologies → Machine learning algorithms**.

KEYWORDS

federated learning; synthetic data; gan; homomorphic encryption; differential privacy

ACM Reference Format:
Ali Reza Ghavamipour, Fatih Turkmen, Rui Wang, and Kaitai Liang. 2023. Federated Synthetic Data Generation with Stronger Security Guarantees. In *Proceedings of the 28th ACM Symposium on Access Control Models and Technologies (SACMAT '23), June 7–9, 2023, Trento, Italy*. ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3589608.3593835

This work is licensed under a Creative Commons Attribution International 4.0 License.

SACMAT '23, June 7–9, 2023, Trento, Italy
© 2023 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0173-3/23/06.
https://doi.org/10.1145/3589608.3593835

1 INTRODUCTION

A Generative Adversarial Networks (GANs) [16] can learn from a set of training data and generate new data with the same characteristics as the training data. GANs are employed in several domains, such as healthcare, energy systems, mobile communications, and finance [42]. Training a GANs over a diverse dataset improves the accuracy of the trained models. However, diverse training data is often distributed across multiple sources and organizations are reluctant to share their data due to legal restrictions over privacy or competition between participants. Therefore, organizations need a way of training GANs in a distributed way so that the whole data set is employed through local training therefore protecting privacy.

For distributed training of GANs, Federated learning (FL) [25] can help. FL is a machine learning technique in which multiple clients collaborate to train a machine learning model without sharing their private data. Federated learning has gradually garnered much attention in both research and industry as a multi-party collaborative machine learning technique. Training GANs in a federated way enables the creation of more accurate and diverse models, as the model is able to make use of a wider range of data from multiple sources that lead to more realistic and diverse generated data. Additionally, federated learning GANs can preserve the privacy of the participating organizations' data, as the data is never shared with any other organization. This can be particularly useful in cases where the data contains sensitive or confidential information.

However, it has been shown that federated learning itself may be subject to privacy issues as communicating model updates throughout the training process can reveal sensitive information [4][26]. By retrieving the shared model updates, an adversary can reconstruct the training data or infer "unintended" information such as membership [37] and property [12]. Moreover, final release of both the discriminator and generator can be the target of such attacks. Chen *et al.* [6] and Hayes *et al.* [17] propose membership inference attack against GANs to determine whether a particular data point was used during the training of the generative model.

To alleviate these privacy issues, it is important to employ appropriate privacy-preserving techniques, such as differential privacy and homomorphic encryption, in the federated learning process. These techniques can protect the privacy of the participating organizations' data while still allowing the model to be trained effectively. Existing solutions for privacy-preserving FL mostly employ differential privacy due to the minimal overhead it introduces. However,

recent work [34] showed that while these models may offer a certain level of privacy, they may also suffer from the poor model utility. They exhibit moderate vulnerability to the membership inference attack when offering an acceptable utility level. Indeed, techniques such as multi-party computing, homomorphic encryption, and Trusted Execution Environment (TEE) strengthening the security guarantees provided by DP are recommended where necessary [20][14].

In this paper, we present a novel method for the asynchronous federated training of a synthetic data generator model with stronger privacy guarantees compared to existing proposals. More specifically, we employ an extension of the FL algorithm introduced by Abadi *et al.* [1] for supporting DP in the federated training of GANs and carefully add CKKS HE [8] on it for achieving stronger privacy guarantees. Thus, the main contributions of this paper are as follows:

- We design a hybrid approach to asynchronous federated generative adversarial training using CKKS homomorphic encryption scheme and differential privacy, which is the first work in this direction to the best of our knowledge.
- We demonstrate that the proposed approach can be utilized to train GANs that offer privacy guarantees and are resistant to multiple potential adversaries.
- We also extensively study the performance of proposed framework with different real-world datasets. We measure the output quality using commonly used GANs quality metrics such as IS, FID, and KIS and compare our proposed approach with other existing studies.

The remainder of the paper is structured as follows: first, in Section 2, we introduce the building blocks of our framework, theoretical properties, and problem descriptions. Then, we will briefly overview the related literature in Section 3. The proposed approach and algorithm will describe in section 4. Our framework is evaluated in Section 5. Section 6 discusses the proposed framework and compares it with existing works. Finally, we give an overview of related work and some concluding remarks.

2 PRELIMINARY

This section introduces the building blocks of our approach and threat model and describes our problem definition.

2.1 Generative Adversarial Networks

The Generative Adversarial Networks (GANs) [16] architecture comprises two deep neural networks, the generator (G) and the discriminator (D), which makes it computationally expensive. The generator and discriminator are closely linked in a traditional GANs setup to achieve the target learning rate. The generator trains itself to generate artificial data, while the discriminator trains itself to differentiate between the original and generated data, and both gradually improve their performance over time. After a certain number of iterations, the generator can produce data that closely resembles the original data. In addition, the discriminator becomes better at identifying the data source. This process can be viewed as a min-max game framework, which the following function can represent:

$$\min_G \max_D = \left[E_{x\sim P_{\text{data}}}\left[\log D_i(x)\right] + E_{z\sim P_z}\left[\log\left(1 - D_i\left(G_i(z)\right)\right)\right]\right] \tag{1}$$

The generator produces synthetic data $G(z)$ and the discriminator outputs a probability $D[0,1]$ of the data being real or fake. The generator aims to minimize the discriminator's ability to identify the source of the data, while the discriminator tries to maximize it.

Moreover, each GANs comprises a discriminator and generator with corresponding parameter vectors $\boldsymbol{\theta}$ and $\boldsymbol{w}$ respectively, loss functions $\mathcal{L}D$ and $\mathcal{L}G$, local true gradients $\boldsymbol{h}(\boldsymbol{\theta},\boldsymbol{w})$ and $\boldsymbol{g}(\boldsymbol{\theta},\boldsymbol{w})$, and local stochastic gradients.

Note that in this work, for a fair comparison with recent works, we mainly consider Deep Convolutional GANs (DCGANs) [33]. The DCGANs architecture is a variant of GANs that employs convolutional neural networks (CNNs) for generating high-quality synthetic images that facilitate the learning of hierarchical features in the images. The generator network of DCGANs learns to generate realistic images by converting random noise inputs into synthetic images, while the discriminator network learns to distinguish between real and synthetic images. The ability of DCGANs to produce highly realistic images has rendered them a preferred choice for image generation applications across diverse domains, including computer vision and graphics.

2.2 Federated Learning

Federated Learning (FL) enables m clients to train a global model $\boldsymbol{w}$ collaboratively without revealing local datasets. Unlike centralized learning, where local datasets have to be collected at a central server before training, FL requires clients to upload the weights of local models ($\{\boldsymbol{w}^i \mid i \in m\}$) to an aggregation server. Given the weights, it aims to optimize the following loss function:

$$\min_{\boldsymbol{w}} \ \ell(\boldsymbol{w}) = \sum_{i=1}^{m} \frac{k_i}{K} L_i(\boldsymbol{w}), L_i(\boldsymbol{w}) = \frac{1}{k_i} \sum_{j\in P_i} \ell_j(\boldsymbol{w}, x_j), \tag{2}$$

where $L_i(\boldsymbol{w})$ and k_i are the loss function and local data size of i-th client respectively. P_i refers to the set of data indices with size k_i.

Without loss of generality and correctness, at t-th iteration, the training of FL using FedAvg [25] algorithm can be divided into four main steps.

• *Global model download.* All connected clients download the global model $\boldsymbol{w}_t$ from the server.

• *Local training.* Each client updates the model parameters through training with their own dataset: $\boldsymbol{w}_t^i \leftarrow \boldsymbol{w}_t^i - \eta \frac{\partial L(\boldsymbol{w}_t, b)}{\partial \boldsymbol{w}_t^i}$, where η and b refer to learning rate and local batch respectively.

• *Aggregation.* After the clients upload local updates $\{\boldsymbol{w}_t^i \mid i \in m\}$, the server can output the global model by averaging them: $\boldsymbol{w}_{t+1} \leftarrow \sum_{i=1}^{m} \frac{1}{m} \boldsymbol{w}_t^i$.

• *Model distribution.* The new global model is sent back to the participating clients, and the process repeats.

The application of classic FL to resource-constrained clients presents various limitations [31]. These include: (1) the unreliability of heterogeneous clients that may go offline unexpectedly, causing the aggregation server to wait for updated local gradients from selected clients; (2) low round efficiency due to the disparity in client

and data heterogeneity, where faster clients must wait for stale local models uploaded from slower clients in each training round; and (3) low resource utilization resulting from inefficient node selection algorithms. To overcome these challenges in this work, asynchronous federated learning (AFL) [31] has been employed, where the server uses a buffer (with the size of K) to store the model updates from each client and aggregates a global model as soon as it collects certain number of local model.

2.3 Homomorphic Encryption

Homomorphic Encryption (HE) is a cryptographic technique that enables users to evaluate (polynomial) computations on ciphertexts without revealing the underlying plaintexts. An encryption scheme is called partial HE if it only supports addition [32] or multiplication [11]. Fully HE [13] on the other hand may support both operations. An HE scheme usually includes the following steps.

• *Key Generation:* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}\ (1^{\lambda})$, where based on the security parameter λ, public key pk and secret key sk are generated.

• *Encryption:* $(c_1, c_2) \leftarrow \mathsf{Enc}(\mathsf{pk}, m_1, m_2)$. By using pk, the probabilistic algorithm Enc encrypts messages m_1, m_2 to ciphertexts c_1, c_2.

• *Homomorphic evaluation:* $\mathsf{Eval}(c_1, c_2) = c_1 \circ c_2 = \mathsf{Enc}(\mathsf{pk}, m_1) \circ \mathsf{Enc}(\mathsf{pk}, m_2) = \mathsf{Enc}(\mathsf{pk}, m_1 \circ m_2)$, where $\circ$ refers to an operator, e.g., addition or multiplication.

• *Decryption:* $m_1 \circ m_2 \leftarrow \mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, m_1 \circ m_2))$. Using sk, the operational results of m_1 and m_2 can be derived.

Our method is based on a variant of fully homomorphic encryption (FHE) scheme called leveled homomorphic encryption (LHE) that supports both addition and multiplication, but only for limited number of times. More specifically, we employ the LHE scheme proposed by Cheon-Kim-Kim-Song (CKKS) [8] that works on an approximation of arithmetic numbers. CKKS, as an emerging encryption scheme, has excellent encryption speed compared to the Paillier and RSA encryption schemes [7]. The CKKS scheme has faster encryption/decryption speed, and supports both additive and multiplicative HE. Similar to the steps in the general homomorphic computations, the CKKS scheme involves the following operations: key generation, encryption, decryption, homomorphic addition, and homomorphic multiplication.

2.4 Differential Privacy

Differential Privacy (DP) [9, 10] is a data protection system tailored to statistical data releases where the privacy of the individuals contributing to the data set is preserved. Suppose we have a dataset $\mathcal{D}$ containing sensitive information that must be made public. To prevent any individual tuple within $\mathcal{D}$ from being easily identifiable, DP is employed. DP involves the use of a randomized algorithm, $\mathcal{A}$, to modify $\mathcal{D}$ in a way that the output produced by $\mathcal{A}$ reveals minimal information about any specific tuple within $\mathcal{D}$. The formal definition of DP is outlined below.

Definition 2.1. **((ϵ, δ) - Differential Privacy)** Given two real positive numbers (ϵ, δ) and a randomized algorithm $\mathcal{A}$: $\mathcal{D}^n \rightarrow \mathcal{Y}$, the algorithm $\mathcal{A}$ provides (ϵ, δ) - differential privacy if for all data sets D, $D^{'} \in \mathcal{D}^n$ differing in only one data sample, and all $\mathcal{S} \subseteq \mathcal{Y}$:

$$Pr[\mathcal{A}(D) \in \mathcal{S}] \leq exp(\epsilon) \cdot Pr[\mathcal{A}(D^{'}) \in \mathcal{S}] + \delta. \tag{3}$$

where ϵ is the privacy budget and δ is the fault-tolerant probability.

We focus on Gaussian noise for differential privacy, as it offers better analysis through Rényi differential privacy framework.

Let x be the input dataset, and let f be a query function. The Gaussian Mechanism adds noise to the output of the query function according to the following formula:

$$\mathcal{A}_{\text{Gauss}}(x, f, \epsilon, \delta) = f(x) + \mathcal{N}\left(\mu = 0, \sigma^2 = \frac{2\ln\left(\frac{1.25}{\delta}\right) \cdot (\Delta f)^2}{\epsilon^2}\right) \tag{4}$$

where $\mathcal{A}$ is a random variable with mean 0 and standard deviation σ such that $\sigma = \frac{S}{\epsilon}$, and ϵ is the privacy budget, a measure of the maximum amount of privacy loss that is acceptable for the dataset.

While the original definition of (ϵ, δ)-DP offers strong data privacy protection, it falls short in addressing privacy leakage resulting from the composition. This issue is also prevalent in federated learning models, where increasing the number of training epochs amplifies privacy leakage. For instance, if a FL client apply ϵ-DP mechanism $\mathcal{A}$ to the gradient before sending it to the central server for k epochs would result in a cumulative privacy loss of $k \times \epsilon$ (due to the composition theorem) at the end of the training.

To address this issue, Mironov [27] introduced Rényi differential privacy (RDP) as a more precise alternative to DP to achieve tighter analysis of composition and amplification methods. Thus, we adopt RDP and its related lemma for the following privacy analysis in this paper.

Definition 2.2. *(Rényi Differential Privacy (RDP)). A randomized mechanism $\mathcal{A}$ with domain $\mathcal{D}$ and range $\mathcal{R}$ satisfies (λ, ε)-Rényi differential privacy with orders $\lambda \in (1, \infty)$ if for any two adjacent inputs $D, D' \in \mathcal{D}$ it holds that:*

$$D_{\lambda}\left(\mathcal{A}(D) \| \mathcal{A}\left(D'\right)\right) = \frac{1}{\lambda - 1} \log E_{\theta \sim \mathcal{A}(D')}\left[\left(\frac{p_{\mathcal{A}(D)}(\theta)}{p_{\mathcal{A}(D')}(\theta)}\right)^{\lambda}\right] \leq \varepsilon$$

where RDP function $D_{\lambda}\left(\mathcal{A}(D) \| \mathcal{A}\left(D'\right)\right)$ is expressed using Rényi divergence, and $p_{\mathcal{A}(D)}(\theta)$ and $p_{\mathcal{A}(D')}(\theta)$ are the densities of $\mathcal{A}(D)$ and $\mathcal{A}\left(D'\right)$, respectively.

In particular, a privacy accounting technique is used to effectively keep track of the (λ, ε) − RDP parameters over the entire range of λ. Then RDP can be converted to standard (ε, δ)-DP for any $\delta > 0$ based on the following Lemma:

The adaptive composition theorem of RDP states that the privacy of a combination of adaptive mechanisms can be determined in relation to the privacy of each individual mechanism. We say a sequence of mechanisms $(\mathcal{A}_1, \ldots, \mathcal{A}_k)$ are chosen adaptively if $\mathcal{A}_i$ can be chosen based on the outputs of the previous mechanisms $\mathcal{A}_1(S), \ldots, \mathcal{A}_{i-1}(S)$ for any $i \in [k]$.

Lemma 2.3. *(Adaptive Composition of RDP [27]). If a mechanism $\mathcal{A}$ consists of a sequence of adaptive mechanisms $(\mathcal{A}_1, \ldots, \mathcal{A}_k)$ with $\mathcal{A}_i$ satisfying (λ, ρ_i)-RDP, $i \in [k]$, then $\mathcal{A}$ satisfies $\left(\lambda, \sum_{i=1}^{k} \rho_i\right)$-RDP.*

Lemma 2.4. *(From RDP to DP[27]). If a randomized mechanism $\mathcal{A}$ satisfies $(\lambda, \varepsilon) - \text{RDP}$, then $\mathcal{A}$ satisfies $\left(\varepsilon + \frac{\log 1/\delta}{\lambda - 1}, \delta\right)$-DP for any $\delta \in (0, 1)$. When $\lambda \to \infty$, RDP converges to $(\varepsilon, 0)$-DP.*

The following lemma demonstrates that privacy is always maintained by a post-processing procedure.

Lemma 2.5. *(Post-processing [27]). Let $\mathcal{A} : \mathcal{Z}^n \to \mathcal{W}_1$ satisfy (λ, ε)-RDP and $f : \mathcal{W}_1 \to \mathcal{W}_2$ be an arbitrary function. Then $f \circ \mathcal{A} : Z^n \to \mathcal{W}_2$ satisfies $(\lambda, \varepsilon) - RDP$.*

2.5 Threat model

In our work, we consider the honest but curious *aggregation servers* and *clients*. In what follows, we summarize the privacy threats posed by each of these actors.

Aggregation Server: The server adheres to the defined protocol (i.e., actively and honestly participates in the training) yet attempts to learn all possible information from legitimately received messages and/or the uploaded models. We consider mainly inference/extraction and reconstruction attacks where the server can calculate the training data of participants unknown to them without colluding with any individual client.

Clients: We assume that the clients may pry at each other's sensitive information to enhance their knowledge by analyzing all the *messages* they receive. The final users of the synthetic data generator are also assumed to be honest but curious, although they do not necessarily participate to the training protocol.

2.6 Problem description

This work investigates generating synthetic data faithful to the original input distribution using GANs framework. The challenge with this setting is that having only one data center may result in limited and homogeneous data, leading to low generalization performance of the model. To overcome this, we look at a scenario that involves multiple data centers with large amounts of data. To allow for collaborative training, the classic FedAvg algorithm in federated settings is used, which exchanges model parameters between the clients and the server. However, this approach does not protect user data from leakage and needs to be improved.

To ensure that the privacy of the original training data is protected against the honest but curious client , the noise will be added to the gradients during the GANs training process. To prove that the GANs preserves privacy, we demonstrate that the generator's parameters guarantee differential privacy concerning the sampled training data. This means that any data generated from the generator will not disclose the privacy of the original training data. To obtain tight privacy bounds, we use the RDP accountant technique, which is a method for measuring privacy loss.

Moreover, using the homomorphic encryption we eliminated the possibility of inference by the honest but curious aggregation server. The encrypted updates are combined to form an encrypted aggregated model, which is then sent back to the parties. The parties then decrypt the model and proceed with the next round of training.

3 RELATED WORKS

Our work is related to two areas: federated GAN and privacy-preserving machine learning. As a result, existing research can be classified into the following two categories:

3.1 Attacks against Federated Learning

Numerous studies, such as [24, 29, 44], have confirmed that even though local datasets are not directly exposed during Federated Learning (FL) training, the uploaded updates (consisting of gradients and weights) can still pose a significant risk to privacy. This risk is exceptionally high if the server is honest-and-curious, as plaintext updates can be retrieved easily on the server side and used to reveal the clients' training data.

Zhu *et al.* [44] proposed a method for training data reconstruction that enables a server to obtain a reconstructed version of the local training dataset of a client in a privacy-preserving manner. The proposed method involves optimizing the Euclidean distance between the uploaded gradients and the gradients trained from dummy samples, which enables the server to retrieve the reconstructed training dataset with high accuracy. To counteract this inference attack, we can utilize homomorphic encryption and differential privacy schemes to ensure that clients can encrypt the updates and send them to the server securely. By using these methods, the updates are protected on the server side since the server does not have sufficient knowledge to perform update reconstruction, which helps to safeguard the privacy of the clients' data.

3.2 Federated Learning GAN

In their study, Rasouli *et al.*[35] proposed FedGAN, which periodically synchronizes local generators and discriminators through an intermediary that averages and broadcasts the parameters. However, since Federated Learning (FL) suffers from non-IID data among clients, Li *et al.*[23] developed a novel framework called SDA-FL. This framework involves each client pretraining a local GANs to generate DP synthetic data, which are then uploaded to the server to construct a global shared synthetic dataset. They also proposed an iterative pseudo-labeling mechanism performed by the server to generate confident pseudo-labels for the synthetic dataset. By combining the local private and synthetic datasets with confident pseudo labels, SDA-FL achieves nearly identical data distributions among clients, improving consistency among local models and benefiting global aggregation.

In addition to optimizing FL-GAN, Xin *et al.* [41] proposed Private FL-GAN to defend against the aforementioned attack. GAN can be securely transmitted through clients and the server using the Lipschitz limit with differential privacy. It is important to note that Private FL-GAN requires the client to transfer the model sequentially during training, whereas the proposed solution allows for parallel GAN training in each round.

Augenstein *et al.*[2] presents a novel federated generative approach that utilizes global differential privacy and a trusted server to preserve data privacy while enabling accurate machine learning on private datasets. The authors discuss the limitations of existing techniques in preserving privacy and maintaining performance in decentralized settings, and introduce their approach that uses DP to address these issues. They provide experimental results to demonstrate the effectiveness of their approach, which allows for accurate machine learning on private datasets while preventing privacy leakage.

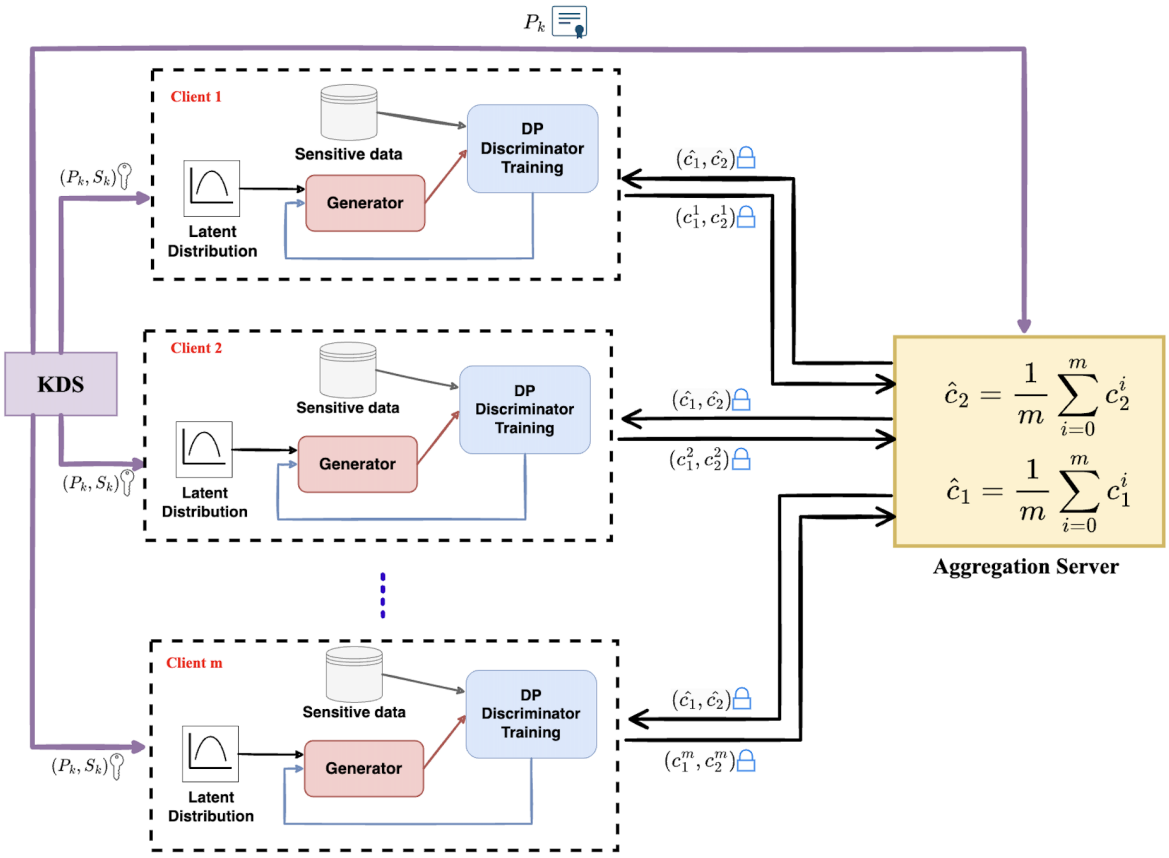

Figure 1: The framework of PP-FedGAN

4 APPROACH

Federated training of a synthetic data generator over sensitive data presents a good alternative to centralized training. However, the global model may seriously compromise privacy [6, 26] once released. The PP-FedGAN framework (Figure 1) presents an asynchronous and (cryptographically) hybrid solution to address possible privacy threats posed by the aggregation server and the clients themselves. In PP-FedGAN, each client encrypts its model updates by using a fast and secure approximate-based CKKS HE scheme before sending it to the aggregation server. The server then takes the average of the (encrypted) updates it receives and broadcasts the results to the clients. Note that we do not need bootstrapping in our method since the number of homomorphic operations is predetermined according to the number of iterations.

For the privacy threats pertinent to clients, we limit the exposure of sensitive information by using DP during local training. More specifically, we adapt a differentially private variant of stochastic gradient descent algorithm (DP-SGD [1]) to GANs, which reduces the impact of individual training examples on the final model. PP-FedGAN meets client-side privacy requirement by guaranteeing the privacy of every sample, not every batch (since these are not meaningful units privacy-wise) in two steps. First, for each sample in the batch, we compute its parameter gradient, and if its norm is larger than the clipping threshold C, we clip the gradient by scaling it down to C. Second, we add random noise from the Gaussian distribution to the gradients before processing.

Our framework consists of three main components:

- **Key Distribution Server (KDS)** to distribute the CKKS Homomorphic Encryption key pairs. We assume the KDS server uses a private channel to distribute the keys to clients.
- **Aggregation Server** broadcasts the average of the received model updates each round.
- **Clients** locally updates the model and sends it to the aggregation server.

We assume that there exists a set of m clients $C_1, C_2, ..., C_m$ that adhere to the agreed GANs structure and with their (disjoint) respective datasets $D_1, D_2, ..., D_m$, and an aggregation server. This means the number of clients is known a priori. We also assume that there are secure channels between clients and the aggregation server. This provides message authentication for incoming messages and prevents an adversary, whether external or a malicious data party, from injecting their own responses.

We consider the following scenario:

Algorithm 1: Server-side training in PP-FedGAN

Input : Ciphertexts received (c_1^i, c_2^i) from k clients such that $k \leq m$ and $i \in \{1, \ldots, k\}$. Training period N, buffer size k and Public Key P_k received from Key Distribution Server

Output : Average of the received inputs $(\hat{c}_1, \hat{c}_2)$

Receive the (P_k) from the KDS

while $n < N$ **do**

  if $n = 0$ **then**

    Loads the initial models $(\theta_0, \boldsymbol{w}_0)$

    Serializes the models $Ser(\theta_0, \boldsymbol{w}_0)$

    Broadcasts them to clients

  else

    while *k updates received from clients* **do**

      Receives (c_1^i, c_2^i)

    end

    Deserializes the encrypted model parameters $\text{Deser}(c_1^i, c_2^i)$

    Computes the average of received data:

    $\hat{c}_1 = \frac{1}{k} \sum_{i=0}^{k} c_1^i \qquad \hat{c}_2 = \frac{1}{k} \sum_{i=0}^{k} c_2^i$

    Serializes $Ser(\hat{c}_1, \hat{c}_2)$

    Broadcasts $Ser(\hat{c}_1, \hat{c}_2)$

  end

  n = n + 1

end

- **Setup:** The KDS creates the keypairs and sends them to the clients. Also, the KDS sends only the public key to the aggregation server.
- **Step 1:** The aggregation server runs the Algorithm 1 and initializes the models (discriminator and generator), and broadcasts them to all the clients after (homomorphically) computing the averages. Next, the server waits for the encrypted model updates from the clients.
- **Step 2:** Clients run the Algorithm 2 in which they receive the global model and update it through several epochs. During the training, differential privacy noise is added to the discriminator model parameters. Next, each layer of the discriminator and generator is encrypted separately. Finally, the encrypted tensors will be serialized since the resulting ciphertexts for the model parameters are too large, and sent to the aggregation server.
- **Step 3:** The aggregation server receives the serialized encrypted model parameters from the clients and store them to the buffer (the buffer size is predefined). If the aggregation server receives enough model updates from the clients, it will deserialize them and compute the average of the encrypted models' parameters by using the client's public key. Finally, the averaged models will be serialized and broadcast to the clients.

At a high level, PP-FedGAN is an asynchronous Federated Generative Adversarial Network that enables differentially private training of GANs in a secure federated setting. It improves the privacy of the GANs training framework by adding random Gaussian noise in the updates of the discriminator. The cumulative privacy loss is tracked by using Rényi Differential Privacy (see Section 2.4). Moreover, the locally trained generator and discriminator models are sent to the honest but curious aggregation server in an encrypted form by each client. The aggregation server then computes the average of the encrypted model parameters and broadcasts the results to clients.

4.1 Formal Security Analysis

The security offered by PP-FedGAN relies on the guarantees provided by the underlying cryptographic techniques, which we formalize below.

Theorem 4.1. *The PP-FedGAN protocol is IND-CPA secure against inference attacks by an honest-but-curious server.*

Proof. Each client in the PP-FedGAN protocol trains their models locally and encrypts the models' parameters (denoted as c_1^i, c_2^i) using the HE CKKS scheme in each round of Algorithm 1. The CKKS scheme involves adding a small error to the message during encryption and relies on the hardness of the Ring learning with errors (RLWE) problem. This scheme is considered to provide IND-CPA security, which is provably an equivalent notion of semantic security [8]. In other words, knowing only ciphertexts, it is infeasible for a computationally-bounded adversary to derive significant information about the plaintexts [15].

In the federated learning setting, the CKKS scheme guarantees the confidentiality and privacy of client model parameters by ensuring that the server cannot learn anything about the model updates of individual clients. The server performs the necessary computations on the encrypted data and sends the encrypted result back to the clients for decryption. By using CKKS, the PP-FedGAN protocol ensures the privacy of each client and guarantees the confidentiality of their model parameters against the honest-but-curious server. □

Theorem 4.2. *The PP-FedGAN protocol provides (ϵ, δ) differential privacy guarantee under the appropriately selected noise scale σ and the clipping threshold C against honest-but-curious clients.*

Proof. During the local training of GANs in PP-FedGAN framework, the gradient of the discriminator undergoes the addition of Gaussian noise with a scale of σ. This ensures that the discriminator satisfies (λ, ε)-RDP, as defined in Definition 2.2, where ε is calculated by the RDP accountant. Because of the post-processing property of differential privacy, as described in Lemma 2.5, the privacy guarantee of PP-FedGAN extends to the generator as well. Specifically, the generator satisfies (λ, ε)-RDP, which can be linearly accumulated over rounds of iterations to obtain a cumulative privacy loss (Lemma 2.3. To ensure stronger privacy guarantees, the RDP privacy parameters are converted to the standard (ϵ, δ)-DP. This is done by applying Lemma 2.4, which converts the (λ, ε)-RDP guarantee to (ϵ, δ)-DP. This provides a strong privacy guarantee, ensuring that the probability of the algorithm leaking any client's information is bounded by δ. Therefore, PP-FedGAN provides a robust privacy framework for training GANs locally, ensuring that clients' data privacy is protected while enabling the generation of high-quality synthetic data. □

Algorithm 2: Client-side training in PP-FedGAN

Input : Initial global GANs model $(\theta_0, \boldsymbol{w}_0)$ or encrypted GANs model $(\hat{c}_1, \hat{c}_2)$ from server S. Training period N. Local training round T. Learning rates of the discriminator (η_D) and generator (η_G).

Output : Updated and encrypted GANs model parameters (c_1, c_2)

(P_k, S_k) (Receive the Public Key and Private Key from the KDS)

while $n < N$ **do**

  if $n = 0$ **then**

    $(\theta_0, \boldsymbol{w}_0)$ (Receives the initial models from the server)

    $\theta = \theta_0 \quad \boldsymbol{w} = \boldsymbol{w}_0$ (Set local GANs model parameters to global model parameters received)

  else

    $(\hat{c}_1, \hat{c}_2)$ (Receives and deserializes the encrypted data)

    $\theta_n = Dec_{Sk}(\hat{c}_1) \quad \boldsymbol{w}_n = Dec_{Sk}(\hat{c}_2)$ (Decrypts the received data)

    $\theta = \theta_n \quad \boldsymbol{w} = \boldsymbol{w}_n$ (Replaces the local GANs discriminator and generator)

  end

  Training discriminator for T rounds:

  $\boldsymbol{g}^{(t)} := \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta} \boldsymbol{w})$ (Compute the per-sample gradients)

  $\hat{\boldsymbol{g}}^{(t)} := \mathcal{A}_{\sigma,C}\left(\boldsymbol{g}^{(t)}\right) = \text{clip}\left(\boldsymbol{g}^{(t)}, C\right) + \mathcal{N}\left(0, \sigma^2 C^2 \boldsymbol{I}\right)$ (Clipping and noise addition)

  $\boldsymbol{\theta}^{(t+1)} := \boldsymbol{\theta}^{(t)} - \eta_D \cdot \hat{\boldsymbol{g}}^{(t)}$ (Gradient descent step)

  Training generator for T rounds:

  $\boldsymbol{w}^{(t+1)} := \boldsymbol{w}^{(t)} - \eta_G \cdot \boldsymbol{h}^{(t)}$ (Gradient descent step)

  $c_1 = Enc_{Pk}(\theta) \quad c_2 = Enc_{Pk}(\boldsymbol{w})$ (Encrypts the discriminator and generator parameters)

  $Ser(c_1, c_2)$ (Serializes the encrypted discriminator and generator)

  Sends the serialized encrypted model parameters to the server S

  n = n + 1

end

5 EXPERIMENTS

This section provides details about implementing the PP-FedGAN protocol and reports on the results of various experiments we conducted[1]. Our experiments include, among others, analysis of the training performance compared to existing work ([40]) and the quality of the generated images given the introduced overhead.

5.1 Experimental setup

PyTorch has been used to implement all algorithms in this work, while TenSEAL [3] and Opacus [43] libraries were employed in implementing homomorphic encryption and differential privacy. All results were obtained on Jupyter Notebooks running on a GPU node with an NVidia V100 GPU, 128GB RAM, and a Linux OS. All the clients used a shared GPU to execute their code during our evaluations.

5.1.1 Framework structures and datasets . Table 1 and 2 present the details of the selected GANs' architecture. The Adam optimizer was utilized with a learning rate of 0.0002 for both the discriminator and generator during training, which lasted for 20 epochs. The Uniform With Replacement Sampler from the Opacus package was employed to sample the training images randomly. The sampling rate was calculated as the batch size of 128 divided by the number of samples, which was 6000 for each client.

We developed our communication system in Python, utilizing ZeroMQ sockets for their low latency and support of atomic multipart messages. This allowed us to integrate components easily and achieve high-throughput in communications. Our implementation was designed to enable intra- and inter-machine communication, making it ideal for facilitating communication between simulated nodes on the same machine or different machines.

Moreover, the CKKS homomorphic encryption layer was implemented using the following parameter values. The value of 8192 was selected as the polynomial modulus, which determines the degree of the polynomials in the ring used for encryption. Additionally, the value of coefficient modulus size was set to [60, 40, 60], where each prime number represents the size of a polynomial modulus. Finally, a precision value of 40 was selected, which refers to the number of bits used to represent each coefficient of the encrypted values. These parameter values were set to the default in all of our experiments involving homomorphic encryption.

We assess the effectiveness of our approaches using three widely used deep learning datasets: MNIST [21], Fashion-MNIST [38], and SVHN [30]. MNIST consists of 60,000 grayscale images of handwritten digits in the training dataset and 10,000 images in the test dataset, each of which is 28x28 pixels. Each digit has 6,000 and 1,000 images in the training and test datasets, respectively. SVHN comprises of 99,289 digits from 10 classes, obtained from house number images in Google Street View. The dataset includes 73,257 digits for training and 26,032 digits for testing, and all digits have been resized to a fixed resolution of 28x28 pixels. Fashion-MNIST consists of 60k images covering ten classes of fashion items at a resolution of 28x28 pixels, which includes 60k training images and 10k test images. We only use the training images in our experiments.

[1]Source code available at https://github.com/gronsec/PP-FedGAN

Layer	Description
Convolution 1	32 filters 4x4
GroupNorm	(64, 64)
Convolution 2	64 filters 4x4
GroupNorm	(64, 128)
Convolution 3	128 filters 4x4
Convolution 4	1 filter 4x4

Table 1: Discriminator architecture

Layer	Description
ConvTranspose 1	128 filters 4x4
GroupNorm	(32, 128)
ConvTranspose 2	64 filters 4x4
GroupNorm	(32, 64)
ConvTranspose 3	32 filters 4x4
GroupNorm	(32, 32)
ConvTranspose 4	1 filter 4x4

Table 2: Generator architecture

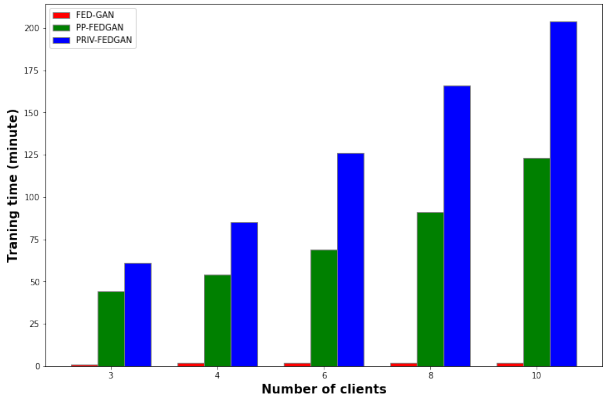

Figure 2: Comparing the effect of increasing number of clients on training federated GANs using different algorithms.

5.2 Evaluation on the Training Performance

We now discuss the communication and computation costs PP-FedGAN introduces during training in terms of the overhead imposed by each layer of protections.

5.2.1 Communication Cost. We assume that the PP-FedGAN framework consists of M clients. In our analysis, we only consider the actual size of θ_M and $\boldsymbol{w}_M$ without taking into account any additional transmission costs incurred by the ciphertext. To further examine the communication costs of the framework, we analyze the costs associated with each round of model aggregation. Specifically, during each round, clients transmit $\text{Enc}(\theta_M, \boldsymbol{w}_M)$ to the aggregation server, and receive $\text{Enc}(\theta_M, \boldsymbol{w}_M)$ from the server. Therefore, the communication costs incurred by the server and each client are $O(Enc(\theta_M, \boldsymbol{w}_M)) + O(Enc(\theta_M, \boldsymbol{w}_M))$.

As the ciphertext constitutes a significant proportion of the communication cost, we will examine the computational cost of the ciphertext in the next subsection.

5.2.2 Computation cost. We conducted an analysis of the computational costs of the proposed protocol, with a focus on the employed HE scheme. As described in Section 2.3, the CKKS approach involves a range of operations, including encryption, ciphertext addition, ciphertext multiplication, and decryption. Table 3 shows the computational costs of ciphertext operations for different sizes of model parameters. This table demonstrates that ciphertexts are considerably larger than plaintexts. Furthermore, the computation cost of model encryption is directly influenced by the number of model parameters. For instance, the encryption time of the discriminator model, which has 109,440 parameters, is 40% faster than the encryption time of the generator model with 312,256 parameters.

5.2.3 Time measurements. The performance of PP-FedGAN is affected by the overhead of multiple security and privacy protection layers. We conducted an experiment with four clients which locally computed the GANs model utilizing a shared GPU unit and communicated with a server over a socket. The time overhead associated with each protection layer and its impact on the overall system performance is shown in Table 4. The first row of the table represents the training time for a naive federated GANs (FEDGAN) implementation without any protection, resulting in efficient training with no extra overhead. However, adding a HE layer to encrypt the trained local model of each client (and to decrypt the received models from the server) increased the training time on the client side by 10 seconds per round. Furthermore, the server needs to perform operations on received ciphertexts to take their average, which results in the most overhead in the whole training process and adds 115 seconds to the entire training procedure. Similarly, adding DP noise during the training increased the FEDGAN training time from 26 to 78 seconds but did not impose any extra overhead on the server. Finally, the last row of the table shows the PP-FedGAN training time. As we expected, combining multiple layers of protection significantly impacted the training procedure. Each round of federated GANs training using the PP-FedGAN architecture took around 3.4 times longer than FEDGAN, increasing the whole training time from 261 to 2096 seconds, which was approximately eight times longer than the naive solution.

Moreover, increasing the number of operations on ciphertext imposes extra computation overhead [28]. In our framework, more operations are needed to be performed by increasing the number of clients. Therefore, it is expected that the total training time will be affected. Figure 2 compares the effect of the increasing number of clients on the total training time of a GANs model using our framework and Priv-FedGAN[40, 41] and FEDGAN. As expected and unlike the FedGAN algorithm, raising the number of clients from 3 to 10 increases the training time of our framework from 43 minutes to 131 minutes. The implementation of a sequential structure in Priv-FedGAN has a notable effect on its execution time, although it does not employ homomorphic encryption. Empirical results indicate that, when the number of clients is set to ten, PP-FedGANshows a performance improvement of up to 36% over Priv-FedGAN.

(a) ϵ = **8.49** **(b)** ϵ = **2.47** **(c)** ϵ = **1.39**

(d) ϵ = **6.2** **(e)** ϵ = **2.1** **(f)** ϵ = **1.3**

Figure 3: Generated images according to different values of privacy parameters on MNIST and Fashion-MNIST datasets.

Model	Number of Parameters	Before Enc. (MB)	After Enc. (MB)	Encryption time (s)
Discriminator	109440	0.432	708.34	6.1
Generator	312256	1.2	708.79	10.2

Table 3: The impact of homomorphic encryption of models on storage and computation.

5.3 Evaluation on generated image quality

This section presents a visualization of the synthetic images generated by our proposed framework, which employs multiple layers of security protection, including homomorphic encryption and differential privacy. We also evaluate PP-FedGAN using commonly used metrics for assessing image-generating models and compare our results to those of other existing work.

5.3.1 Visual quality comparison. We conducted experiments on MNIST and Fashion-MNIST datasets to illustrate the relationship between the privacy level and the quality of output images of the generator. Additionally, since the CKKS scheme is an approximate homomorphic encryption scheme (i.e., RLWE), it can introduce some approximation errors in the computation. Therefore, it is crucial to evaluate the quality of the output generated by the PP-FedGAN framework, particularly the image quality. To further ensure the quality of the synthetic data, the PP-FedGAN framework was tested with different levels of differential privacy parameters and with the default values of the HE parameters of the CKKS scheme described in Section 5.1.1.

The synthetic data generated by the framework is presented in Figure 3, along with the corresponding selected values of DP parameters for each image. Table 5, provides the selected training parameters corresponding each generated image in Figure 3. The clipping parameter for all datasets was set to a constant value of 1.0 in all experiments. To ensure reliable results, each GAN trained with additional noise was trained five times.

As shown in Figure 3, the DP noise multiplier parameter directly affects the image quality. As the noise multiplier increases, the value of ϵ decreases, resulting in lower-quality output. Particularly, when the noise multiplier is increased from 0.5 to 1.0, the value of ϵ

	Each round (Client-side)	Each round (Server-side)	10 rounds training
FedGAN	26	-	261
FedGAN + HE	36	115	1531
FedGAN + DP	78	-	784
PP-FedGAN	89	115	2096

Table 4: Training time overhead of each protection layer (seconds).

Figure 3 sub-image	a	b	c	d	e	f
Epsilon ϵ	8.49	2.47	1.39	6.2	2.1	1.3
Clipping norm	1.0	1.0	1.0	1.0	1.0	1.0
Noise multiplier	0.5	0.7	1.0	0.5	0.7	1.0
Batch size	128	128	128	128	128	128
Targeted δ	1e-4	1e-4	1e-4	1e-4	1e-4	1e-4

Table 5: The different privacy regimes (ϵ, δ) with the corresponding hyperparameters

decreases, leading to a reduction in image quality in all the images. This can be observed in the decreased sharpness and resolution of the images as the noise multiplier increases. Therefore, choosing the appropriate value for ϵ parameter is crucial for balancing the trade-off between image quality and privacy protection in synthetic data generation using PP-FedGAN.

5.3.2 Quality comparison using evaluation metrics. Evaluation of generative models, including GANs, is a challenging task and needs to capture image diversity and quality. In a perfect world, evaluating the quality of generated data would require human judgment. As a result, promising approaches to mimicking this human judgment are being developed in the field of GANs.

One of the most popular metrics is the Inception Score (IS) proposed by Salimans et al. [36] and has been demonstrated to be closely linked to human judgment. The IS works by passing the generated data through a pre-trained Inception classifier, where a high score indicates diverse and high-quality images, while a low score suggests uniform and low-quality images. However, the Inception model may struggle to capture significant feature variation from GANs trained on MNIST and SVHN datasets. To achieve more reliable results on the MNIST dataset, we adapted the LeNet [22] model instead of the Inception model to calculate these scores.

In addition to IS, we also utilize the Fréchet Inception Distance (FID) [18] and Kernel Inception Distance (KID) [5] metrics. The FID measures the distance between the generated image distribution and the real data. It utilizes the Inception network to compare both sets of data and assumes that their outputs have Gaussian distributions. Similarly, the KID score measures the dissimilarity between two probability distributions based on random samples from each distribution, using the Inception network. The KID score is impartial and thus more reliable, especially when there are fewer test images than the Inception features' dimension. Lower FID and KID scores suggest better regenerated input images, indicating the model's ability to preserve the original quality. To evaluate performance on the MNIST dataset, we use the LeNet [22] model instead of the Inception model to calculate these scores.

To assess the effect of privacy budgets on the performance of PP-FedGAN, we conducted experiments using different noise multiplier values ranging from 0.1 to 2, which correspond to different values of ϵ in the Opacus library. By evaluating our model under different levels of privacy protection, we can determine the optimal privacy budget that allows us to generate high-quality synthetic images while ensuring the privacy and security of the training data.

Furthermore, we compared the quality of the generated images by PP-FedGAN with the closest works to ours, Priv-FedGAN [41] and DP-GAN [39] frameworks. By adapting their algorithm to our experimental setup, we were able to compare their performance with that of PP-FedGAN and determine which framework performed best under different privacy budgets.

Strict privacy budgets lead to a rise in the Inception score, indicating a decline in the quality of the synthesized samples. Figure 4 (a) and (b) represent the IS score for SVHN and MNIST datasets, respectively. We gradually increase the noise multiplier value from 0.1 to 2.0, which corresponds to a reduction in the epsilon value, the Inception score for the images produced by PP-FedGAN on the MNIST dataset decreases from 6.2 to 2.21, while for images generated on the SVHN dataset, the Inception score decreases from 7.01 to 1.53. In comparison, Priv-FedGAN performs marginally better than PP-FedGAN, and DP-GAN achieved the lowest score among the three training algorithms.

Figures 4 (c) and (d) illustrate the FID and KID scores for MNIST data using three different privacy-preserving algorithms. Gradually increasing the noise multiplier value from 0.1 to 2.0 causes the PP-FedGAN's FID score to increase from 2.32 to 73.73. In other words, the quality of the generated images decreases gradually when the noise multiplier value exceeds 1.5. In this experiment, DP-GAN achieves a highly similar score to PP-FedGAN. Meanwhile, Priv-FedGAN generates images with slightly lower quality than the other algorithms.

Similarly, for the KID score, FED-DGAN and PP-FedGAN outperform the Priv-FedGAN algorithm. The FED-DGAN achieves a score of 0.026, PP-FedGAN achieves 0.053, and Priv-FedGAN only manages 0.25. As expected, the performance of all generator models decreases rapidly as the privacy budget shrinks.

Also, the Figure 4 (c) and (d) corresponding to the FID and KID scores over MNIST data. By slowly increasing the noise multiplier's value from 0.1 to 2.0, FID score increases from 2.32 to 73.73 and KID score from 0.053 to 2.264. As we expected, the performance of all models rapidly deteriorates as the privacy budget becomes smaller. When the privacy budget is significant, our model generates higher-quality images.

6 DISCUSSION

The PP-FedGAN framework is designed to generate synthetic data using an asynchronous federated learning approach in a privacy-preserving manner. In our framework (same as Augenstein *et al.*'s work [2]), the server-based FedAVG is selected as the federated learning algorithm. Unlike PP-FedGAN, Priv-FedGAN uses a sequential federated learning approach, with DP used to ensure privacy guarantees. Sequential training can be more prone to overfitting than parallel training because the model is only trained on a single client's data. As a result, the model may learn to perform well

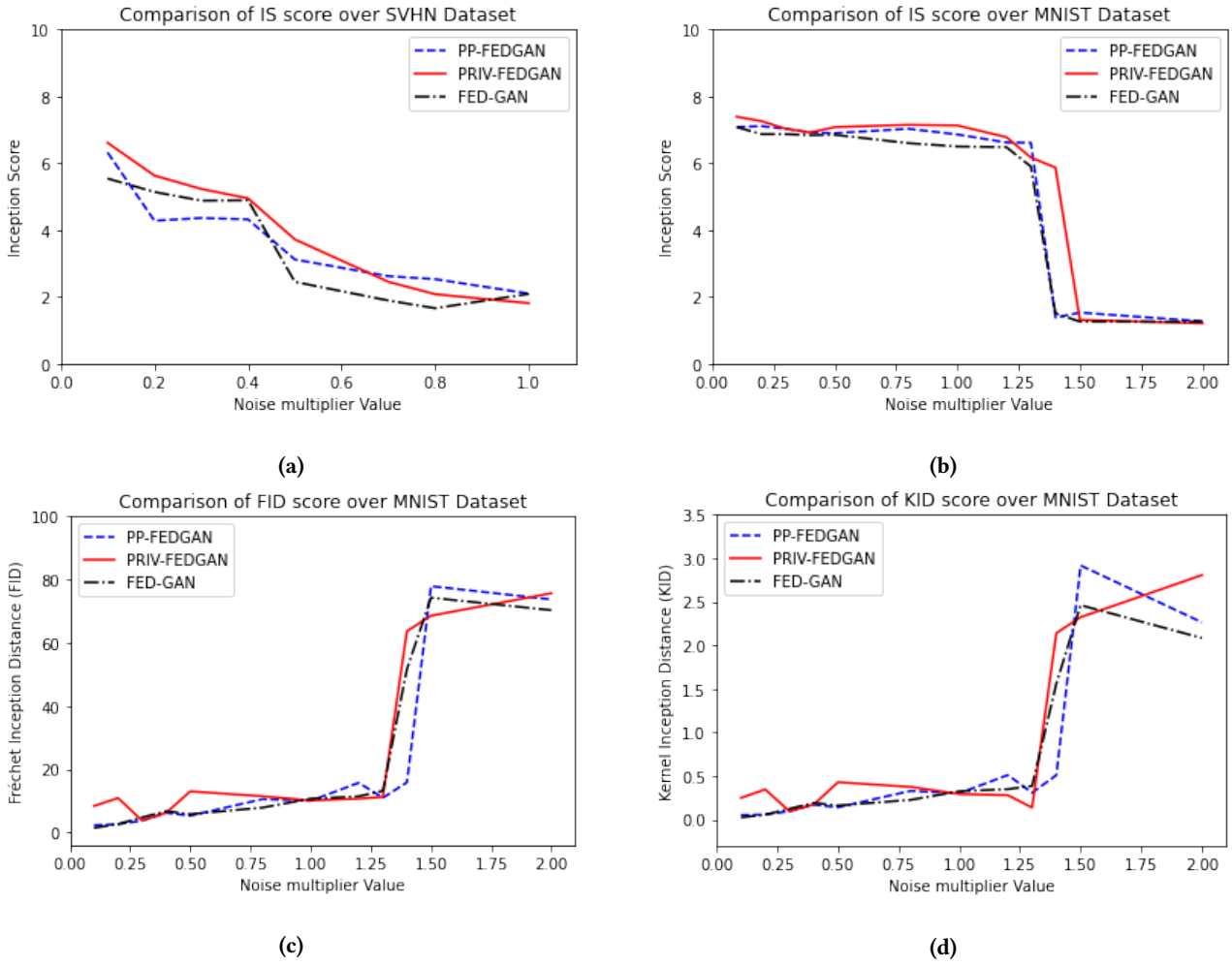

Figure 4: Comparison of evaluation metrics on SVHN and MNIST datasets

on the training data from that specific client but may need to generalize better to other clients. Also, as the results of our experiments expressed, training can take longer to converge to a solution than parallel training because the model must be trained on each client's data in sequence. This can result in longer overall training times and thus less fault tolerance, which can be a significant drawback for larger datasets or when working with many clients.

Moreover, the PP-FedGAN framework incorporates local differential privacy (DP) and homomorphic encryption (HE) to provide privacy guarantees at both local and global levels. Local DP ensures privacy during training, while HE prevents the server from accessing the model updates. In contrast, Augenstein *et al.*'s approach employs global differential privacy, which depends on a trusted aggregation server to ensure privacy. This can potentially result in reduced privacy protection due to the possibility of sensitive data leakage [19].

Overall, the PP-FedGAN framework provides a solid approach to privacy-preserving synthetic data generation, with several unique advantages over existing methods. Using HE and DP in a parallel federated learning setting, in conjunction with the separate training of the generator and discriminator models, allows for greater control over the quality of the synthetic data generated while providing strong privacy guarantees.

7 CONCLUSION

In this paper, we proposed a novel privacy-preserving federated learning scheme (PP-FedGAN) based on the asynchronous FedAVG algorithm that utilizes homomorphic encryption and differential privacy. Using DP, we can guarantee overall privacy from the inference of any model output of our framework and available intermediate results. Additionally, employing HE guarantees that any messages exchanged between the client and server are not revealed and therefore do not leak private information.

Given these guarantees, the generative model produced by our system achieved up to 30% better training performance than existing works such as Priv-FedGAN. Additionally, we conducted experiments on three public datasets, and the experimental results showed that our framework can generate high-quality synthetic

data. We achieved the inception score of 6.9 when the ϵ value was 8.49 compared with 6.85 for DP-GAN.

For future work, we are planning to consider Byzantine adversaries, which can send arbitrary incorrect messages to the primary server with the aim of data corruption, communication failure, or malicious attacks, thereby deviating from the learning model.

REFERENCES

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*. 308–318.

[2] Sean Augenstein, H Brendan McMahan, Daniel Ramage, Swaroop Ramaswamy, Peter Kairouz, Mingqing Chen, Rajiv Mathews, et al. 2019. Generative models for effective ML on private, decentralized datasets. *arXiv preprint arXiv:1911.06679* (2019).

[3] Ayoub Benaissa, Bilal Retiat, Bogdan Cebere, and Alaa Eddine Belfedhal. 2021. TenSEAL: A library for encrypted tensor operations using homomorphic encryption. *arXiv preprint arXiv:2104.03152* (2021).

[4] Abhishek Bhowmick, John Duchi, Julien Freudiger, Gaurav Kapoor, and Ryan Rogers. 2018. Protection against reconstruction and its applications in private federated learning. *arXiv preprint arXiv:1812.00984* (2018).

[5] Mikołaj Bińkowski, Danica J Sutherland, Michael Arbel, and Arthur Gretton. 2018. Demystifying mmd gans. *arXiv preprint arXiv:1801.01401* (2018).

[6] Dingfan Chen, Ning Yu, Yang Zhang, and Mario Fritz. 2020. Gan-leaks: A taxonomy of membership inference attacks against generative models. In *Proceedings of the 2020 ACM SIGSAC conference on computer and communications security*. 343–362.

[7] Hao Chen, Wei Dai, Miran Kim, and Yongsoo Song. 2019. Efficient multi-key homomorphic encryption with packed ciphertexts with application to oblivious neural network inference. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 395–412.

[8] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. 2017. Homomorphic encryption for arithmetic of approximate numbers. In *International conference on the theory and application of cryptology and information security*. Springer, 409–437.

[9] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. 2006. Our data, ourselves: Privacy via distributed noise generation. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 486–503.

[10] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*. Springer, 265–284.

[11] Taher ElGamal. 1985. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE transactions on information theory* (1985), 469–472.

[12] Karan Ganju, Qi Wang, Wei Yang, Carl A Gunter, and Nikita Borisov. 2018. Property inference attacks on fully connected neural networks using permutation invariant representations. In *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*. 619–633.

[13] Craig Gentry. 2009. *A fully homomorphic encryption scheme*. Stanford university.

[14] Ali Reza Ghavamipour, Fatih Turkmen, and Xiaoqian Jiang. 2022. Privacy-preserving logistic regression with secret sharing. *BMC Medical Informatics and Decision Making* 22, 1 (2022), 1–11.

[15] Shafi Goldwasser and Silvio Micali. 1982. Probabilistic Encryption and How to Play Mental Poker Keeping Secret All Partial Information. In *Proceedings of the 14th Annual ACM Symposium on Theory of Computing, May 5-7, 1982, San Francisco, California, USA*, Harry R. Lewis, Barbara B. Simons, Walter A. Burkhard, and Lawrence H. Landweber (Eds.). ACM, 365–377.

[16] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2020. Generative adversarial networks. *Commun. ACM* 63, 11 (2020), 139–144.

[17] Jamie Hayes, Luca Melis, George Danezis, and Emiliano De Cristofaro. 2017. Logan: Membership inference attacks against generative models. *arXiv preprint arXiv:1705.07663* (2017).

[18] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. 2017. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems* 30 (2017).

[19] Briland Hitaj, Giuseppe Ateniese, and Fernando Perez-Cruz. 2017. Deep models under the GAN: information leakage from collaborative deep learning. In *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*. 603–618.

[20] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. 2021. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning* 14, 1–2 (2021), 1–210.

[21] Yann LeCun. 1998. The MNIST database of handwritten digits. *http://yann. lecun. com/exdb/mnist/* (1998).

[22] Yann LeCun et al. 2015. LeNet-5, convolutional neural networks. *URL: http://yann. lecun. com/exdb/lenet* 20, 5 (2015), 14.

[23] Zijian Li, Jiawei Shao, Yuyi Mao, Jessie Hui Wang, and Jun Zhang. 2022. Federated learning with gan-based data synthesis for non-iid clients. *arXiv preprint arXiv:2206.05507* (2022).

[24] Xinjian Luo, Yuncheng Wu, Xiaokui Xiao, and Beng Chin Ooi. 2021. Feature inference attack on model predictions in vertical federated learning. In *ICDE*. 181–192.

[25] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *AISTATS*. 1273–1282.

[26] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. 2019. Exploiting unintended feature leakage in collaborative learning. In *2019 IEEE symposium on security and privacy (SP)*. IEEE, 691–706.

[27] Ilya Mironov. 2017. Rényi differential privacy. In *2017 IEEE 30th computer security foundations symposium (CSF)*. IEEE, 263–275.

[28] Michael Naehrig, Kristin Lauter, and Vinod Vaikuntanathan. 2011. Can homomorphic encryption be practical?. In *Proceedings of the 3rd ACM workshop on Cloud computing security workshop*. 113–124.

[29] Milad Nasr, Reza Shokri, and Amir Houmansadr. 2019. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In *IEEE S&P*. 739–753.

[30] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. 2011. Reading Digits in Natural Images with Unsupervised Feature Learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*. http://ufldl.stanford.edu/housenumbers/nips2011_housenumbers.pdf

[31] John Nguyen, Kshitiz Malik, Hongyuan Zhan, Ashkan Yousefpour, Mike Rabbat, Mani Malek, and Dzmitry Huba. 2022. Federated learning with buffered asynchronous aggregation. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 3581–3607.

[32] Pascal Paillier. 1999. Public-key cryptosystems based on composite degree residuosity classes. In *EUROCRYPT*. 223–238.

[33] Alec Radford, Luke Metz, and Soumith Chintala. 2015. Unsupervised representation learning with deep convolutional generative adversarial networks. *ICLR* (2015).

[34] Md Atiqur Rahman, Tanzila Rahman, Robert Laganière, Noman Mohammed, and Yang Wang. 2018. Membership Inference Attack against Differentially Private Deep Learning Model. *Trans. Data Priv.* 11, 1 (2018), 61–79.

[35] Mohammad Rasouli, Tao Sun, and Ram Rajagopal. 2020. Fedgan: Federated generative adversarial networks for distributed data. *arXiv preprint arXiv:2006.07228* (2020).

[36] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. 2016. Improved techniques for training gans. *Advances in neural information processing systems* 29 (2016).

[37] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. 2017. Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*. IEEE, 3–18.

[38] Han Xiao, Kashif Rasul, and Roland Vollgraf. 2017. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747* (2017).

[39] Liyang Xie, Kaixiang Lin, Shu Wang, Fei Wang, and Jiayu Zhou. 2018. Differentially private generative adversarial network. *arXiv preprint arXiv:1802.06739* (2018).

[40] Bangzhou Xin, Yangyang Geng, Teng Hu, Sheng Chen, Wei Yang, Shaowei Wang, and Liusheng Huang. 2022. Federated synthetic data generation with differential privacy. *Neurocomputing* 468 (2022), 1–10.

[41] Bangzhou Xin, Wei Yang, Yangyang Geng, Sheng Chen, Shaowei Wang, and Liusheng Huang. 2020. Private fl-gan: Differential privacy synthetic data generation based on federated learning. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2927–2931.

[42] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. 2019. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)* 10, 2 (2019), 1–19.

[43] Ashkan Yousefpour, Igor Shilov, Alexandre Sablayrolles, Davide Testuggine, Karthik Prasad, Mani Malek, John Nguyen, Sayan Ghosh, Akash Bharadwaj, Jessica Zhao, et al. 2021. Opacus: User-friendly differential privacy library in PyTorch. *arXiv preprint arXiv:2109.12298* (2021).

[44] Ligeng Zhu, Zhijian Liu, and Song Han. 2019. Deep leakage from gradients. *NIPS* (2019).