Achieving Data Reconstruction Hardness and Efficient Computation in Multiparty Minimax Training

Truong Son Nguyen
Arizona State University
Tempe, AZ, USA
snguye63@asu.edu

Yi Ren
Arizona State University
Tempe, AZ, USA
yiren@asu.edu

Guangyu Nie
Arizona State University
Tempe, AZ, USA
gnie1@asu.edu

Ni Trieu
Arizona State University
Tempe, AZ, USA
ntrieu1@asu.edu

Abstract

Generative models have achieved remarkable success in a wide range of applications. Training such models using proprietary data from multiple parties has been studied in the realm of federated learning. Yet recent studies showed that reconstruction of authentic training data can be achieved in such settings. On the other hand, multiparty computation (MPC) guarantees standard data privacy, yet scales poorly for training generative models. In this paper, we focus on improving reconstruction hardness during Generative Adversarial Network (GAN) training while keeping the training cost tractable. To this end, we explore two training protocols that use a public generator and an MPC discriminator: Protocol 1 (P1) uses a fully private discriminator, while Protocol 2 (P2) privatizes the first three discriminator layers. We prove reconstruction hardness for P1 and P2 by showing that (1) a public generator does not allow recovery of authentic training data, as long as the first two layers of the discriminator are private; and through an existing approximation hardness result on ReLU networks, (2) a discriminator with at least three private layers does not allow authentic data reconstruction with algorithms polynomial in network depth and size. We show empirically that compared with fully MPC training, P1 reduces the training time by 2× and P2 further by $4-16\times$. Our implementation can be found at https://github.com/asu-crypto/ppgan.

Keywords

secure machine learning, minimax training, multiparty computation, generative adversarial neural network

1 Introduction

Background. Generative models have shown significant success in applications from content generation [34], automated programming [70] to scientific discoveries [21]. To achieve emergent intelligence and generalization, these models often rely on large aggregated datasets [47]. While data augmentation has been used to bootstrap the training [48], there exist industry settings where multiple providers of proprietary datasets agree to collaborate on building a generative model, yet would like to be assured that their own data is kept private during the training process. In practice, we expect the resultant generative model, i.e., the *generator*, to be public to all participants so that inferences can be done efficiently, and also to mimic the true data distribution so that it is qualified to perform downstream tasks. For these reasons, the training of such generative models does not enjoy standard security definitions that are rooted in indistinguishability likelihoods (e.g., cryptographic and differential privacy). Indeed, as shown in [5], targeted extraction of training data from a public generative model is possible. Therefore, the focus of this paper is to achieve *reconstruction hardness*: we design model architectures and training protocols such that untrusted servers that execute the training protocol cannot successfully reconstruct sensitive training data during or after the training even in the worst-case scenarios. Without loss of generality, we will focus on protocols for solving minimax problems, i.e., generative adversarial network (GAN), and our method can be applied to minimization problems, e.g., diffusion models [61], which we discuss in Sec. B. The research question we address is the following:

Does there exist a GAN training protocol that achieves both reconstruction hardness and tractable computation?

Problem formulation. We introduce the following settings to formalize the problem.

Training protocol: Before training, each data holder sends an additive secret share [14] of their data to all of the two computing servers, who then use these shares to together update a generator and a discriminator following a protocol. Depending on the protocol, the generator and the discriminator can be either secret-shared (or "private"), partially private, or public, during the training process. Regardless of the protocol, the generator will be made public after the training.

Threat model: We consider a semi-honest model following [4, 18, 29, 54] wherein servers are incentivized to adhere to the training protocol but may try to reconstruct authentic data from public information released during and after the training. This adversarial goal is similar to the Data Reconstruction attack in [57], and we refer to their work for further discussion on the relation between this goal and other types of attack on training, such as Membership Inference and Attribute Inference.

Proceedings on Privacy Enhancing Technologies 2025(3), 44–60

https://doi.org/10.56553/popets-2025-0088

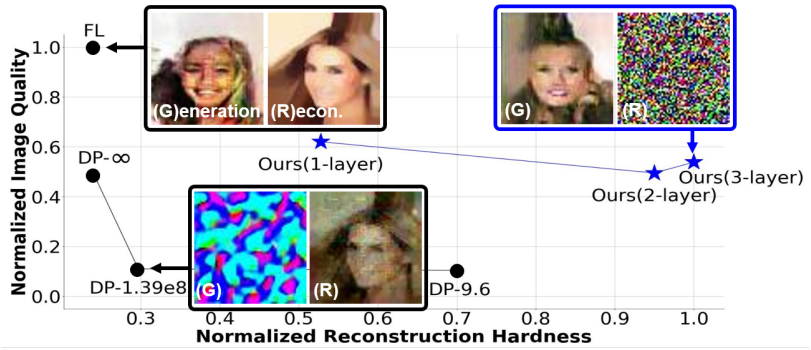


Figure 1: Reconstruction hardness (R) versus image generation quality (G) for federated learning (FL), differential privacy (DP), and the proposed protocols (P2 with 1 to 3 private discriminator layers). The x-axis represents the normalized inverse of CW-SSIM for reconstructed images (higher means better privacy), and the y-axis represents the normalized inverse of FID for the resultant generators (higher means better quality). DP-∞, DP-1.39e8, and DP-9.6 are DPs with corresponding ϵ values. On CelebA, our method (P2 with 2 private layers) is 8× slower than DP GAN training and 6× faster than full MPC training, while preserving reconstruction hardness and image generation quality.

Servers are unable to change their (either private or public) inputs or outputs. This paper considers a two-server non-colluding setting. Let training data be $X = \{X_t\}_{t=1}^{T}$, where X_t is used for each training iteration $t \in [T]$. Server i follows a protocol $\mathcal{M}$ and gain information $\mathcal{M}(X_t)$ at each iteration. $\mathcal{M}(X_t)$ may include (i) the parameter trajectories of the public generator and discriminator layers throughout the training, (ii) the learning rates used for model updates, and (iii) the public discriminator outputs corresponding to generated inputs. In the analysis, we consider the worst case that favors the attacker, where $X_t \in X$ contains a single data point and $\mathcal{M}$ is deterministic.

Reconstruction hardness: A reconstruction $\hat{X}_t$ is considered successful if "sensitive" information about X_t is revealed. Therefore reconstruction hardness is a data-dependent notion: A successful reconstruction could be the recognition of generic data categories, or the revealing of specific text labels on images. Without a consensus on its definition, this paper defines reconstruction hardness based on the L2 distance $\rho(\cdot,\cdot) : \mathbb{R}^{d_x} \times \mathbb{R}^{d_x} \to \mathbb{R}$ as follows:

Definition 1.1. $\mathcal{M}$ achieves (ϵ, δ)-*reconstruction hardness* if there does not exist an algorithm with polynomial data and computational complexities that, with probability at least $1-\delta$, computes $\hat{X}_t$ based on $\mathcal{M}(X_t)$ such that $\rho(\hat{X}_t, X_t) \leq \epsilon$ for some $t \in [T]$.

In our empirical evaluation on image generation, we use standard image quality metrics CW-SSIM [58], PSNR [27], and Feat-MSE [19] to define and compare hardness. Two images are more similar if their CW-SSIM and PSNR are high, or Feat-MSE is low. See Section 2.6 for definitions of these metrics.

Contributions. Our contributions are three-fold:

(1) We propose two simple (unconditional or traditional) GAN training protocols that rely on the MPC framework: Protocol 1 (P1) uses a fully private discriminator, and Protocol 2 (P2) privatizes the first three layers of the discriminator. Both P1 and P2 use public generators. Additionally, we then extend these protocols to support conditional GANs, offering an efficient approach to training data-secured classifiers and addressing tasks that require data labeling.

(2) We provide the first proof on reconstruction hardness for P1 and P2 by showing that (1) publicizing the generator still achieves reconstruction hardness, as long as the first two layers of the discriminator are private; and through an existing approximation hardness result on ReLU networks, (2) a ReLU discriminator with at least three private layers achieves reconstruction hardness even if its remaining layers are public. This is the primary contribution of our work.
In general, it is assumed that all computations in ML algorithms must remain private when implemented using secret-sharing (via MPC) or encrypted settings (via homomorphic encryption). However, to the best of our knowledge, no prior work has specifically explored the security implications of applying MPC to GAN training. Through our proof, we demonstrate the level of security guarantees that can be achieved when a few layers of computation are made public to enhance efficiency.

(3) We show empirically that compared with a full MPC implementation of our unconditional GAN training, P1 reduces the training time by 2× and P2 further by 4 to 16×; and compared with DP implementations, our protocols achieve better reconstruction hardness. To the authors' best knowledge, this is one of the first methods that tractably train GAN models on CelebA with a reconstruction hardness proof.
For conditional GANs, our primary focus is on classification accuracy, as their runtime and communication cost is almost similar to those of the corresponding unconditional protocols. To evaluate this, we train a multilayer perceptron (MLP) and a logistic regression (LR) model on downstream classification task using images generated by our conditional privacy-preserving GAN models and compare it to state-of-the-art secure GS-WGAN [8], and show that our solution provide data that help classification model improve the performance by 2-3% accuracy.

Difference from standard privacy. It is necessary to clarify the differences between our privacy definition, i.e., (ϵ, δ)-reconstruction hardness, and existing ones. First, standard definitions are rooted in distinguishability of likelihoods, where privacy is considered achieved when the difference between the likelihoods of $\mathcal{M}(X)$ and

$\mathcal{M}(X')$ is within some threshold. For example, (ϵ, δ)-DP uses the divergence of the likelihood between two adjacent datasets [16]. As is discussed in [64], limited likelihood difference alone does not explain how much of the training set X can be reconstructed by an adversary either during or after the training. To this end, [64] introduced probably approximately correct (PAC) privacy, which follows Def. 1.1. Our threat model for hardness analysis, however, is critically different from that of PAC privacy: PAC privacy assumes $\mathcal{M}$ to be known to the attacker so that the output distribution of $\mathcal{M}(X)$ can be computed; in this study, $\mathcal{M}$, e.g., the computation of discriminator and generator gradients, is at least partially unknown due to the private layers of the discriminator. We rule out two potential attempts to circumvent this difficulty for an attacker: First, one may consider these private layers as part of the input data to be reconstructed, yet their prior distribution at an arbitrary training step t is unknown, which is a prerequisite for PAC privacy. Second, one may attempt to approximate the distribution of $\mathcal{M}(X)$ by passing a large batch X through a blackbox $\mathcal{M}$. Yet this can be prevented by the training protocol that constrains the size and content of X processed by the servers. To summarize, our privacy definition focuses on reconstruction hardness of a specific training setting rather than distinguishability of likelihoods, and is thus different from DP variants; and our threat mode takes into account the practical reconstruction challenges introduced by private layers and the training protocol, and is thus different from PAC privacy. Essentially, we exchange one type of tradeoff for privacy, namely, the lack of generation quality caused by gradient noise introduced in DP and PAC privacy, for another type of tradeoff, namely, the additional cost of secure computation on private layers. Fig. 1 summarizes the performance tradeoffs of all training protocols.

Notations. We will use D_t (resp. G_t) as the discriminator (resp. generator) at training iteration t, and use $\theta_{D_t} \in \mathcal{H}_D$ (resp. $\theta_{G_t} \in \mathcal{H}_G$) as its parameters, with $\mathcal{H}_D$ and $\mathcal{H}_G$ being the hypothesis spaces. We use α_t (resp. β_t) as the learning rate of the discriminator (resp. generator) updates, and $\nabla_\theta D$ (resp. $\nabla_\theta G$) as the model gradient with respect to its parameters.

2 Related Work

2.1 Generative Adversarial Network (GAN)

GANs [24] represent a class of minimax algorithms for training generative models, i.e., models that approximate the mapping of random samples z from a standard distributions p_z to a data distribution p_x. A GAN is composed of a generator $G : \mathbb{R}^{d_z} \to \mathbb{R}^{d_x}$ that synthesizes data and a discriminator $D : \mathbb{R}^{d_x} \to [0, 1]$ that assesses the authenticity of the generated data. The training solves a Nash equilibrium between G and D:

$$\min_G \max_D \quad L(D, G) = \mathbb{E}_{x\sim p_x}\left[\log D(x)\right] + \mathbb{E}_{z\sim p_z}\left[\log(1 - D(G(z)))\right]. \tag{1}$$

This adversarial interplay between the generator and discriminator constitutes the core dynamics of GANs, enabling the matching between p_G and p_x. Recent advancements of GANs [25, 30, 59] have propelled the development of various privacy sensitive applications such as medical imaging [35, 60, 67], networking and server traces [37] and facial image generation [33, 46]. The deployment of these applications has raised concerns over private information leakage since the generated samples of a GAN can reflect its underlying training dataset's property and potentially disclose privacy-sensitive data. Our paper is not concerned about post-hoc treatments of generative models to prevent the accidental generation of authentic data, but rather about the training protocols to prevent reconstruction of sensitive data during multiparty training.

There is a variant of traditional GAN which is Conditional GAN (CGAN) [41]. Unlike GAN, CGAN allows user to specify the constraint in which the generated image should reflect. For example, a CGAN on MNIST dataset allows users to specify which digit that the generated image is classified as. Thus, CGAN provides an effective way to generate synthetic data for downstream task such as training a data-secured classifier.

2.2 Secure and Decentralized GAN Training

Federated Learning (FL) has been used to train GANs with multiple clients for the sake of data privacy. For example, FedGAN [53] achieves image generation quality comparable to normal GANs. [2, 8] take differential privacy (DP) into account by adding random noises to the training gradients, which causes slow convergence in solving the minimax problem. AsyncGAN [6] introduces a secure training protocol for conditional GANs where multiple data holders train their own discriminators and a centralized server trains a generator by querying gradient information from the discriminators. This protocol assumes that the data distributions of all data holders are known to the central server. Using MNIST as an example, the server knows that a particular data holder holds images of "0"s, and will only send "0"-labeled images to its discriminator. We do not use this assumption since it leaks data type information to the server. Instead, our protocols train a single generator and a discriminator without any knowledge about the training dataset.

In addition to existed approach, there is a naive approach applying MPC [31, 44] in a straightforward manner, similar to the ways MPC is applied in training machine learning model [31]. We refer to this approach as Protocol P0 in our paper, and try to optimize the performance of it using some weaker security assumption of reconstruction hardness.

Finally, one can also implement normal GAN training augmented by defenses against data reconstruction attack. [39] proposes and tests several defense mechanism against data reconstruction attack. Their paper finds that Gradient Pruning, with high probability of weight being pruned, achieves the best defense against data reconstruction. However, the reconstructed image quality shown in their work still shows limitation of the defense, as the structure of the real input is still being preserved. We show a comparison with such a defense later in Section 5.3 where our protocol achieves a far better reconstruction defense compared to [39].

2.3 Multiparty Computation

We review multiparty computation (MPC) as it plays a central role in our solution. MPC is a method for parties to evaluate any arbitrary functions without revealing any information about the input data. MPC can have any number of data holders as long as they secretly share their data to a specific number of servers who later do the computation. A conventional MPC framework consists

n servers and m data holders participating in a two-stage process to compute a function f:

- **Step 1:** Data-holder breaks the data x into additive pieces $[\![x]\!]_{i\in[n]}$. The pieces sum up to the original data: $\sum_{i=1}^{n} [\![x]\!]_i = x$. Then, they distribute the shares one piece per server, i.e. $[\![x]\!]_i$ to server $\mathcal{S}_i$.
- **Step 2:** Servers use the additive shares to perform secure computation to get $f(x) = C([\![x]\!]_1, \ldots, [\![x]\!]_n)$ where $C(\cdot)$ is an interactive computation process on the additive shares.

We note that if the training is executed fully in MPC, then standard security is achieved in the sense no information about the training data is leaked to the servers since each only hold a random shares of the data, unless all servers collude. In order to securely evaluate any functions using additive piece of the user data, MPC often relies on three core operations: Addition, Multiplication, and Comparison. We show the details of how each operation works in Section 2.5. In this paper, private layers of the discriminator will use MPC for training. As a result, no servers have any knowledge about the weights and biases of these layers, nor do they know the input to and outputs from these layers.

Multiparty computation in machine learning. Privacy-preserving machine learning (PPML) [15, 17, 28, 36, 42, 43, 45, 49, 51, 55] allows different entities to collaboratively and privately train and evaluate machine learning models using their collective data. The existing body of literature on PPML has applications to linear regression, logistic regression, neural networks, and transformers [36, 38, 50, 56]. Most PPML schemes operate in a server-aided setting, where data owners delegate the computation to a small number of servers that are neither trusted nor colluding. [45] introduced the first practical PPML system based on a two-server setting. Designs employing three servers [42] and four servers [51] offer a weaker security guarantee, as collusion between any pair of these servers can reveal the private data of the data owners. Consequently, the two-server PPML model remains preferable. Despite satisfying standard cryptographic security, MPC leads to significantly higher training cost. This paper follows the server-aided framework using two non-trusted and non-colluding servers. But instead of following the standard security/privacy definition, we use reconstruction hardness as a data-dependent privacy interpretation, which allows us to hybridize private and public layers to achieve more tractable training.

2.4 Vertical Federated Learning (VFL)

Works such as VFL-GAN [68] leverages federation of clients to jointly train a GAN model. In such a scheme, clients pass the data through a local model, then send the intermediate output to a central server to combine and do further computation. Even though the results of such a model is promising, the use cases of VFL-GAN and MPC-GAN are different. VFL-GAN having clients participate in training process has a bottleneck on low power client or client with poor communication and goes offline often. The probability of having this kind of client increases when the number of clients increase. Thus, in use cases where there are many clients, VFL might not be optimal. MPC, on the other hand, does not require clients to be involved in the training process and is competitive in that use case. The main focus of this paper is on improving the performance of secure GAN in the MPC use case.

2.5 MPC Implementation

We implement training protocols using Crypten [31], a PyTorch framework for multi-party computation. Here we describe in detail some of the core operations that we extensively used. MPC hyperparameters are chosen as Crypten default values, which are results of their extensive experiments for values with good efficiency/accuracy trade-off [32].

Addition. To compute $[\![x + y]\!]$, each parties compute $[\![x]\!]_i + [\![y]\!]_i$ and the share is $[\![x + y]\!]_i = [\![x]\!]_i + [\![y]\!]_i$.

Multiplication. To compute $[\![x * y]\!]$, all parties pre-exchange shares of a Beaver triple (a, b, c) such that $ab = c$. Then they reveal value of $\epsilon = x+a, \delta = y+b$. Then the final output $[\![xy]\!] = \epsilon * [\![y]\!]_i - \delta * [\![a]\!]_i + [\![c]\!]_i$.

Comparison. To compute $[\![x < y]\!]$, the parties either implement a garbled circuit [66] or convert $[\![x - y]\!] = [\![x]\!] - [\![y]\!]$ to a "binary" version $\langle x - y\rangle$ such that $\oplus\langle x - y\rangle = x - y$ and evaluate the first bit of $\oplus(\langle x - y\rangle)$ [32].

Data type conversion. Crypten uses integer in a group $\mathbb{Z}_q$ for secure computation. Thus, it uses a *scale-and-round* algorithm for converting floating-point numbers to integers. In particular, to convert a floating-point number $x \in \mathbb{R}$ to an integer $x_c \in \mathbb{Z}_q$, Crypten multiplies x by a large number B and round it to the nearest integer:

$$x_c = \lfloor Bx \rceil.$$

In order to convert x_c back to x, Crypten simply divides it by B:

$$x = \frac{x_c}{B}.$$

In our implementation, we set $B = 2^{16}$ and $q = 2^{64}$.

Additive sharing. We define $[\![x]\!]$ as the sharing of a private value x. In the two-server setting, to securely distribute an ℓ-bit secret value x, the data owner randomly chooses $[\![x]\!]_1$ in the arithmetic field $\mathbb{Z}_{2^\ell}$, computes $[\![x]\!]_2 = x - [\![x]\!]_1$, and sends each $[\![x]\!]_i$ to a server $S_{i\in[2]}$. To reconstruct a secret x, each server sends its shared $[\![x]\!]_i$ to the data owner who computes $x = [\![x]\!]_1 + [\![x]\!]_2$. When sharing a set of values $\{x_1, \ldots, x_n\}$, the data owner can employ Pseudorandom Generator (PRG) to generate $[\![x_i]\!]_1$ from a PRG seed s. Consequently, only the PRG seed s needs to be transmitted to S_1 instead of all $[\![x_i]\!]_1$. This approach significantly enhances communication efficiency. In our implementation, $l = 64$.

Multiplication. Crypten uses Beaver's trick to perform secure multiplication. This is described in Section 2.5.

Comparison. Crypten first let servers compute $[\![x - y]\!] = [\![x]\!] - [\![y]\!]$ and convert it to the binary share version $\langle x - y\rangle$ which has the property: $\oplus\langle x - y\rangle = x - y$. The share of comparison $[\![x > y]\!]$ is then just the first bit of $\langle x - y\rangle$

Exponentiation and Sigmoid. Sigmoid is calculated easily if we have exp approximation. In Crypten, exp approximation can be done in several ways. In our implementation, we use the approach using iterations which is based on the equation:

$$\lim_{n\to\infty} (1 + \frac{x}{2^n})^{2^n}$$

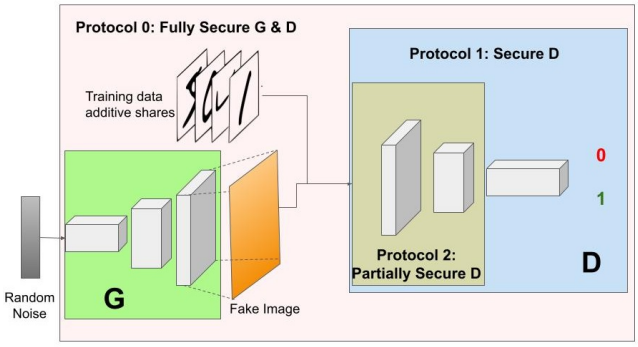


Figure 2: Schematics of the Proposed Protocols

In our implementation, we set $n = 8$.

Logarithm. Similar to exponentiation, Crypten has many methods for approximating log function. We choose the approach based on high-order modified Householder method, which to compute $\log(x)$ it has the update rules as:

$$h_n = 1 - x\exp(-y_n)$$

,

$$y_{n+1} = y_n - \sum_{k=1}^{\text{ord}} \frac{1}{k} h_n^k$$

In our experiment, we choose $n = 2, \text{ord} = 8$ and number of iterations for calculating exp is 8.

Leaky-ReLU. To enable secret-shared computation of Leaky ReLU, we rewrite its standard formula $\max(0, x) + a\min(0, x)$ as an MPC-friendly formula $x(\text{sign}(x) + a - \text{sign}(x)a)$ which is a combination of one addition, one multiplication-by-a-constant (which can be done locally without any communication), one secure comparison, and one secure multiplication.

2.6 Image Reconstruction Metrics

CW-SSIM. [58] The Complex-Wavelet Structural Similarity function is defined as:

$$\text{CW-SSIM}(c_x, c_y) = \left(\frac{2\sum_{i=1}^{N}|c_{x,i}||c_{y,i}|+K}{\sum_{i=1}^{N}(|c_{x,i}|^2+|c_{y,i}|^2)+K}\right)\left(\frac{2|\sum_{i=1}^{N} c_{x,i}c_{y,i}^*|+K}{2\sum_{i=1}^{N}|c_{x,i}c_{y,i}^*|+K}\right),$$

where c_x, c_y represent complex wavelet transform of the images x and y respectively; K be small positive number, ideally set to 0.

PSNR. [27] Peak signal-to-noise ratio (PSNR) function is defined as:

$$\text{PSNR}(x, y) = 20\log_{10}(\max(x)) - 10\log_{10}(\text{MSE}(x, y)),$$

where MSE represents the mean square error function.

Feat-MSE. [20] Given a model $\mathcal{M}$, the feat MSE is defined as the mean square error between two outputs, formally:

$$\text{Feat-MSE}(x, y) = \text{MSE}(\mathcal{M}(x), \mathcal{M}(y)),$$

In our paper, we use the Discriminator D as the model $\mathcal{M}$.

3 Proposed Learning Architecture and Protocols

This section presents two GAN architectures and training protocols, each comprising four phases, see Alg. 3. In Phase I, every data owner secretly shares their private dataset $X^{(i)}$ among servers. Phase I is communication-optimized using a pseudorandom generator (PRG) [22] (details in Section 2.5). Since P1 and P2 use the same Phase I implementation, we only report computational cost of Phases II to IV. Figure 2 shows the high-level design of our protocols.

3.1 Protocol 0 (P0): Fully-secure MPC

A vanilla solution is to directly apply MPC to GAN training, in which case all intermediate values, such as layer-wise forward and backward outputs, are secret-shared among the servers. P0 enjoys standard security, assuming the security of MPC, and serves as the baseline for comparisons on computational cost and generation quality. P0 is implemented using Crypten [31].

3.2 Protocol 1 (P1): Private Discriminator

Based on the observation that generator training only involves random samples rather than authentic training data, we propose P1 where only the discriminator is kept private. We prove in Sec. 4.3 that reconstruction hardness is achieved in P1.

At a high level, our protocol P1 assumes that the generator model G is a public and common model to all the servers, while the discriminator model D is kept secret by the MPC mechanism. Our P1 first begins with **Data Sharing** step, similar to P0, where data holder $\mathcal{D}_j$ additively shares his data to all the servers. Second, each of the servers samples a random noise vector z, and computes $\tilde{X} = G(z)$. Then, the server additively shares $\tilde{X}$ to $n-1$ other servers. With secret shares of $\tilde{X}, X, D$, n servers use MPC protocol to compute the binary cross entropy loss as $\text{dloss} = -\mathbb{E}[\log D(X)] - \mathbb{E}[\log(1-D(\tilde{X})]$ and get the gradient $\nabla_{\theta_D}\text{dloss}$ to update the weights θ_D of discriminator. Then, $\mathcal{S}_1$ computes $\tilde{X} = G(z)$, broadcasts that value to all servers. Now, all server compute and reveal the value of $\text{grad} = \nabla_{\tilde{X}}(-\mathbb{E}[\log(D(\tilde{X}))]$. This gradient value will be then used to calculate $\nabla_{\theta_G}\text{gloss} = \text{grad} \cdot \nabla_{\theta_G}\tilde{X}$. Note that $\nabla_{\theta_G}\tilde{X}$ is public to all servers as G and $\tilde{X}$ are public.

3.3 Protocol 2 (P2): Partially-private Discriminator

We propose P2 to further reduce training cost. Here only the first few layers of the discriminator are private, after which, the private outputs from both servers are combined and made public. The remainder of the discriminator forward pass is done locally on an individual server without the need for communication or secure computation. We prove in Sec. 4.4 that reconstruction hardness can still be achieved for a ReLU network.

At a high level, protocol P2 differs from P1 in the third step: **Discriminator Pass**. In P2's **Discriminator Pass**, we divide D into two parts: D_{pri} which is kept as additive shares across servers, and D_{pub} which is publicly shared to all servers. The servers after use MPC to compute $[\![\tilde{y}]\!] = [\![D_{pri}(\tilde{X})]\!]$ and $[\![y]\!] = [\![D_{pri}(X)]\!]$, will then reveal these intermediate values. Later, the discriminator loss and the corresponding gradient is publicly computed. We need

PARAMETERS: m data holders $\{\mathcal{D}_1, \ldots, \mathcal{D}_m\}$ with corresponding data $\{X^{(1)}, \ldots, X^{(m)}\}$; n computing servers $\{\mathcal{S}_1, \ldots, \mathcal{S}_n\}$; Generator model G; Discriminator model D

PROTOCOL:

I. **Data Sharing**

(1) Each data-holder $\mathcal{D}_i$ secretly break their data $X^{(i)}$ into n additive shares $[\![X^{(i)}]\!]_{j\in[n]}$

(2) $\mathcal{D}_i$ sends $[\![X^{(i)}]\!]_{j\in[n]}$ to computing server $\mathcal{S}_j$

For each training iteration, n servers $\{\mathcal{S}_1, \ldots, \mathcal{S}_n\}$ together perform the following:

II. **Generator Forward**

(1) Each $\mathcal{S}_{i\in[n]}$ chooses a random noise vector z, compute $\tilde{X} = G(z)$

(2) Each $\mathcal{S}_{i\in[n]}$ locally breaks $\tilde{X}$ into n pieces and sends $[\![X]\!]_j$ to $\mathcal{S}_j$

III.a. **Discriminator Pass** (Private Discriminator, P1)

(1) n server use MPC to

(a) Calculate dloss $= -\mathbb{E}[\log D(X)] - \mathbb{E}[\log(1 - D(\tilde{X}))]$

(b) Update Discriminator parameters using Gradient Descent on gradient ∇_{θ_D}dloss

(2) $\mathcal{S}_1$ compute $\tilde{X} = G(z)$, broadcasts $\tilde{X}$ to all $\mathcal{S}_i$

(3) n servers use MPC to calculate grad $= \nabla_{\tilde{X}}(-\mathbb{E}[\log(D(\tilde{X})])$

III.b. **Discriminator Pass** (Partially Private Discriminator, P2)

(1) n servers together reveal $\tilde{y} = D_{pri}(\tilde{X})$, $y = D_{pri}(X)$

(2) Each server $\mathcal{S}_i$ locally do:

(a) Update D_{pub} parameters using Gradient Descent on gradient $\nabla_{\theta_{D_{pub}}}$dloss $= (\nabla_{\theta_{D_{pub}}} - \mathbb{E}[\log D_{pub}(y)] - \mathbb{E}[\log(1 - D_{pub}(\tilde{y}))])$

(b) Update D_{pri} parameters using Gradient Descent on gradient $(\nabla_{\theta_{D_{pri}}} y \cdot \nabla_y + \nabla_{\theta_{D_{pri}}} \tilde{y} \cdot \nabla_{\tilde{y}})$dloss

(3) $\mathcal{S}_1$ compute $\tilde{X} = G(z)$, broadcasts $\tilde{X}$ to all $\mathcal{S}_i$

(4) n servers use MPC to calculate grad $= \nabla_{\tilde{X}}(-\mathbb{E}[\log(D_{pub} \circ D_{pri}(\tilde{X}))])$

IV. **Generator Update**

(1) Each server $\mathcal{S}_i$ do:

(a) Receive $[\![\text{grad}]\!]_i$ from (III.a) or (III.b)

(b) Calculate $[\![\nabla_{\theta_G}\text{gloss}]\!]_i = n\alpha \cdot [\![\text{grad}]\!]_i \cdot \nabla_{\theta_G}\tilde{X}$

(c) Update G using Gradient Descent to G_i using $[\![\nabla_{\theta_G}(-\mathbb{E}[\log(D(\tilde{X})])]\!]_i$

(2) n servers together get true updated value $G = \frac{1}{n}\sum_{i=0}^{n} G_i$

Figure 3: Our Unconditional Privacy Preserving Generative Adversarial Network (PPGAN)

to compute the gradient of the private weight $\nabla_{\theta_{D_{pri}}}$ using MPC based on the value of ∇_ydloss and $\nabla_{\tilde{y}}$dloss

3.4 Conditional GAN Training

Conditional GAN [41] provides an effective way to train a data secured classifier and tasks that require data label. We provide a protocol, which adapt from unconditional GAN, but with an additional MPC component: label encoding. First, in the **Data Sharing** step, the data holder $\mathcal{D}_i$ embeds the label $y^{(i)}$ into an embedding $y^{(i)}_{emb} = \text{embed}(y^{(i)})$ where embed is an embedding function such as one hot encoding. Then, he breaks that embedding into n additive shares and sends them to n servers. In **Generator Forward** step, each server $\mathcal{S}_{i\in[n]}$ samples a fake label $\tilde{y}$ uniformly along with the random noise vector z sampled from a Normal distribution, then from $[z : \tilde{y}_{emb}]$ (where $[\cdot : \cdot]$ indicates concatenation) the server $\mathcal{S}_i$ generates a fake image $\tilde{X}$ corresponding to the fake label $\tilde{y}$. The server $\mathcal{S}_i$ then converts the fake image $\tilde{X}$ and fake embeded label $\tilde{y}$ into n additive shares and distributes the shares to all servers. Next, in **Discriminator Pass** step, n servers use MPC to calculate the share of the loss and the share of the gradient, as well as update the weight of the discriminator using the gradient. n servers also use MPC to compute the gradient with respect to generator output grad $= \nabla_{\tilde{X}}$gloss. All servers finally reconstruct the value of grad and use it in **Gradient Update** step to update the weight of the generator using gradient descent. We show the detail of Conditional GAN training algorithm in Figure 4.

3.5 Correctness of Our protocols

We note that P1 and P2 update the generator based on secret shares of the discriminator. Let α_t be the generator learning rate, and $[\![\theta_{G_t}]\!]_i$ follows a gradient descent update: $[\![\Delta\theta_G]\!]_i = [\![-\alpha_t \nabla_{\theta_G} L]\!]_i$. Then since the secret shares are additive, it is easy to show that $\theta_{G_{t+1}} = \sum_i [\![\theta_{G_t}]\!]_i + [\![\Delta\theta_G]\!]_i$. Note that in practice, secret shares introduce small numerical errors, which affect the training convergence.

4 Reconstruction Hardness

In this section, we first lay out our assumption on the threat model. Second, we introduce Proposition 1, which justifies the generality of Def. 1.1 when applied to image generation, i.e., reconstruction hardness defined on all these metrics can be alternatively defined through ρ (the proof in Section 4.2). Therefore, our hardness analyses will be based on Def. 1.1. Finally, we provide a detail analysis on the security of our protocol P1 and P2 for unconditional image generation, the security proofs for corresponding conditional generation protocols can be similarly derived.

PARAMETERS: m data holders $\{\mathcal{D}_1, .., \mathcal{D}_m\}$, n computing server $\{\mathcal{S}_1, .., \mathcal{S}_n\}$, a generator network G, a discriminator network D, an Embedding function emb that maps the label $y \in [10]$ to a 10×1 vector.
PROTOCOL

I. **Data Sharing**
(1) For each $i \in [m]$ do
(a) Data holder $\mathcal{D}_i$ embed all the label $y^{(i)}$ to $y^{(i)}_{emb} = \text{emb}(y^{(i)})$
(b) Data holder $\mathcal{D}_i$ breaks their data $X^{(i)}$, $y^{(i)}_{emb}$ into n additive secret shares $[\![X^{(i)}]\!]_{j\in[n]}$ such that $\sum_{j=1}^{n} [\![X^{(i)}]\!]_j = X^{(i)}$, $\sum_{j=1}^{n} [\![y^{(i)}_{emb}]\!]_j = y^{(i)}_{emb}$.
(c) $\mathcal{D}_i$ sends $[\![X^{(i)}]\!]_j, [\![y^{(i)}_{emb}]\!]_j$ to computing server $\mathcal{S}_j$ for $j \in [m]$

II. **Generator Forward**
(1) For each $\mathcal{S}_i \in \{\mathcal{S}_1, ..\mathcal{S}_n\}$ do
(a) Choose a random noise vector z, and a label $\tilde{y}$
(b) Get the embedded label vector $\tilde{y}_{emb} = \text{emb}(\tilde{y})$
(c) Get fake data $\tilde{X}^{(i)} = G([z : \tilde{y}_{emb}])$ where $[\cdot : \cdot]$ indicates concatenation of 2 vectors.
(d) Break $\tilde{X}^{(i)}$ and $\tilde{y}^{(i)}_{emb}$ to n pieces $[\![\tilde{X}^{(i)}]\!]_{j\in[n]}$ and $[\![\tilde{y}^{(i)}_{emb}]\!]$
(e) Send $[\![\tilde{X}^{(i)}]\!]_j, [\![\tilde{y}_{emb}]\!]_j$ to $\mathcal{S}_j$

III. **Discriminator Pass** (Private Discriminator GAN, Protocol 1)
(1) Update discriminator: The n servers use MPC to
(a) Calculate dloss $= -\mathbb{E}[\log D([X : y_{emb}])] - \mathbb{E}[\log(1 - D([\tilde{X} : \tilde{y}_{emb}]))]$
(b) Update Discriminator parameters using Gradient Descent on gradient ∇_{θ_D} dloss
(2) Calculate Generator loss:
(a) Use only 1 party $\mathcal{S}_i$ to get $\tilde{X} = G([z : \tilde{y}_{emb}])$
(b) n servers use MPC to calculate gloss $= -\mathbb{E}[\log D(\tilde{X})]$
(c) n servers use MPC to calculate grad $= \frac{\partial}{\partial \tilde{X}}$gloss

III.b. **Discriminator Pass** (Partially-private Discriminator GAN, Protocol 2)
(1) Update discriminator: The n servers use MPC to
(a) n servers together reveal $\tilde{t} = D_{pri}([\tilde{X} : \tilde{y}_{emb}]), t = D_{pri}([X : y_{emb}])$
(b) Calculate dloss $= -\mathbb{E}[\log D_{pub}(t)] - \mathbb{E}[\log(1 - D_{pub}(\tilde{t}))]$
(c) Update D_{pub} parameters using Gradient Descent on gradient $\nabla_{\theta_{D_{pub}}}$ dloss
(d) Update D_{pri} parameters using Gradient Descent on gradient $(\nabla_{\theta_{D_{pri}}} t \cdot \nabla_t + \nabla_{\theta_{D_{pri}}} \tilde{t} \cdot \nabla_{\tilde{t}})$dloss
(2) Calculate Generator loss:
(a) Use only 1 party $\mathcal{S}_i$ to get $\tilde{X} = G([z : \tilde{y}_{emb}])$
(b) n servers locally calculate gloss $= -\mathbb{E}[\log D_{pub}(\tilde{t})]$ where $\tilde{t} = \text{reveal}(D_{pri}([\tilde{X} : \tilde{y}_{emb}]))$
(c) n servers use MPC to calculate grad $= \frac{\partial}{\partial \tilde{X}}$gloss

IV. **Generator Update**
(1) Each $\mathcal{S}_i$ receives $[\![\text{grad}]\!]_i$ from PPGAN.DiscriminatorPass
(2) $\mathcal{S}_i$ calculates $[\![\nabla_{\theta_G}\text{gloss}]\!]_i = n\eta \cdot [\![\text{grad}]\!]_i \cdot \nabla_{\theta_G}\tilde{X}$
(3) $\mathcal{S}_i$ updates G using Gradient Descent to G_i using $[\![\nabla_{\theta_G}\text{gloss}]\!]_i$
(4) n parties together get true updated value $G = \frac{1}{n}\sum_{i=0}^{n} G_i$

Figure 4: Our Conditional Privacy Preserving Generative Adversarial Network (Conditional PPGAN)

4.1 Threat Model

To justify the protocol design, we consider the following worst-case assumptions that favor an attacker: (1) The attacker has access to all public information during and after the training, including the terminal generator and its distribution p_{G_T}; (2) p_{G_T} matches the true training data distribution p_x; (3) the discriminator uses a sigmoid output activation and the discriminator loss is a cross-entropy function; (4) a single data point $x_t \sim p_x$ is drawn at iteration t and used to update D_t via gradient descent, where the learning rate α_t is public. For private layers of the discriminator, Assumptions (3) and (4) lead to the following update:

$$[\![\theta_{D_t}]\!] = [\![\theta_{D_{t-1}}]\!] - \alpha_t [\![\nabla_\theta D_{t-1}(x^t)/D_{t-1}(x^t)]\!]. \tag{2}$$

We assume updates on one data point. As shown empirically in [20], reconstructing one data point is easier for an attacker than a batch. Then with an artificial sample $\tilde{x} := G_{t-1}(\tilde{z})$ where $\tilde{z} \sim p(z)$, the generator is updated based on D_t. In P1 where D_t is private, an aggregation of secret shares of $D_t(\tilde{x})$ and $\nabla_x D_t(\tilde{x})$ is needed before computing the following update:

$$\Delta\theta_{G_t} = -\alpha_t(1 - D_t(\tilde{x}))^{-1}\nabla_x D_t(\tilde{x})\nabla_\theta G_{t-1}(\tilde{z}). \tag{3}$$

Since $\tilde{z}$ is known, Eq. (2) and Eq. (3) together define a deterministic mechanism from $(\theta_{D_{t-1}}, x^t)$ to $\Delta\theta_{G_t}$, which we denote by $\mathcal{M}(\cdot,\cdot) : \mathcal{H}_D \times \mathcal{X} \to \mathcal{H}_G$.

We examine reconstruction hardness of the proposed architecture and training protocol in this section. To overview, we consider two types of attacks that exploit the public information in these protocols: The first type (Sec. 4.3) uses the fact that the public update of the generator is based on that of the discriminator, which is in turn based on the authentic data. We show that the public

update of the generator achieves reconstruction hardness as long as the discriminator has at least two private layers. The second type (Sec. 4.4) concerns data reconstruction through public outputs of the discriminator when only the first few discriminator layers are private. Here we leverage existing approximation hardness results to show that with at least three private layers, authentic training data cannot be recovered with polynomial data and computational complexity.

4.2 Connection between Reconstruction Hardness and Image Reconstruction

The following Proposition (Proposition 1) gives a connection between L2 distance and image similarity distance metrics such as CW-SSIM, PSNR, FMSE. This connection shows that we can use a bound in terms of the L2 distance to conservatively satisfy bounds in terms of these three similarity metrics. Therefore our theoretical analysis uses the L2 distance only. For empirical results in Section 5.3, we report results using all three similarity metric.

Proposition 1. *Let ρ_1, ρ_2, and $\rho_3 : \mathbb{R}^{d_x} \times \mathbb{R}^{d_x} \to \mathbb{R}$ be CW-SSIM, PSNR, and FMSE metrics. For any x and $y \in \mathbb{R}^{d_x}$ such that $\rho(x,y) \le \epsilon$, there exist c_1, c_2, $c_3 > 0$, such that $\rho_1(x,y) \ge \max\{1 - c_1\epsilon, 0\}$, $\rho_2(x,y) \ge 20(c_2 - \log_{10}\epsilon)$, and $\rho_3(x,y) \le c_3\epsilon$.*

Proof. **For CW-SSIM:** We start by considering two signals x and $y \in \mathbb{R}^{d_x}$ that lead to complex wavelet coefficients $c_{y,i} = c_{x,i} + \Delta c_i \in \mathbb{C}$ for $i = [M]$ where M is the number of discrete wavelet coefficients. Let $\alpha = [|c_{x,i}|]_{i=1}^M$ and $\beta = [|\Delta c_i|]_{i=1}^M$ where $|\cdot|$ is the modulus. Since α and β are vectors with non-negative elements, we have $\beta = \Lambda\alpha$ where $\Lambda = diag([\lambda_i])$ is positive definite. Let $\lambda = \max_i \lambda_i$. Note that λ is bounded if $\alpha_i \neq 0$ for all $i \in [M]$ which is a reasonable assumption to make for e.g., a natural image x. CW-SIMM for c_y and c_x is then

$$\begin{aligned} S(c_y, c_x) &= \frac{2|\sum_i c_{x,i}c_{y,i}^*| + K}{\sum_i |c_{x,i}|^2 + |c_{y,i}|^2 + K} \\ &= \frac{2|\sum_i |c_{x,i}|^2 + c_{x_i}\Delta c_i^*| + K}{\sum_i 2|c_{x,i}|^2 + |\Delta c_i|^2 + 2Re(c_{x,i}\Delta c_i^*) + K} \\ &\ge \frac{2\left(\sum_i |c_{x,i}|^2 - |c_{x_i}||\Delta c_i|\right)}{\sum_i 2|c_{x,i}|^2 + |\Delta c_i|^2 + 2|c_{x,i}||\Delta c_i|} \end{aligned} \tag{4}$$

$$\begin{aligned} S(c_y, c_x) &= \frac{2\alpha^T(\alpha - \beta)}{\alpha^T\alpha + (\alpha+\beta)^T(\alpha+\beta)} \\ &\ge \frac{2(1-\lambda)}{1 + (1+\lambda)^2}. \end{aligned}$$

Since this lower bound of CW-SSIM monotonically decreases with respect to λ, we need to find an upper bound of λ to lower bound S. To do so, we first denote the Fourier transform of x and y as $X(w)$ and $Y(w)$ with frequency w, respectively; $\Delta(w) = Y(w) - X(w)$; $G(w)$ the Fourier transform of the wavelet filter; w_c the center frequency; and (s_i, p_i) the ith discrete scaling and translation factors. Here we use j as the imaginary unit. Then we have

$$\lambda = \max_i \frac{\left|\int_{-\infty}^{\infty} \Delta(w)G(s_i w - w_c)e^{jwp_i}dw\right|}{\left|\int_{-\infty}^{\infty} X(w)G(s_i w - w_c)e^{jwp_i}dw\right|} \le k|\Delta| \tag{5}$$

where $|\Delta| = \max_w|\Delta(w)|$ and $k = \max_i \frac{\left|\int_{-\infty}^{\infty} G(s_i w - w_c)e^{jwp_i}dw\right|}{\left|\int_{-\infty}^{\infty} X(w)G(s_i w - w_c)e^{jwp_i}dw\right|}$.

Lastly, let $\Delta x(u) = y(u) - x(u)$ be the spatial domain difference. We have

$$\begin{aligned} |\Delta|^2 &= \max_w \left|\int_{-\infty}^{\infty} \Delta x(u)e^{-i2\pi wu}\right|^2 \\ &\le 2\left(\int_{-\infty}^{\infty} \Delta x(u)du\right)^2 \\ &\le 4\rho^2(x,y). \end{aligned} \tag{6}$$

Therefore if $\rho(x,y) \le \epsilon$, $S(c_y, c_x) \ge \frac{2(1-2k\epsilon)}{1+(1+2k\epsilon)^2}$. For small ϵ, we have $S(c_y, c_x) \ge 1 - 4k\epsilon$.

For PSNR: Given x and $y \in \mathbb{R}^{d_x}$ and $\rho(x,y) \le \epsilon$, we have the following upper bound on PSNR:

$$\rho_2(x,y) = 20\log_{10}\max x - 10\log_{10}\rho^2(x,y) \le 20\log_{10}\max x - 20\log\epsilon. \tag{7}$$

For FMSE: Given x and $y \in \mathbb{R}^{d_x}$, and let $c^{(l)} : \mathbb{R}^{d_x} \to \mathbb{R}^{d_l}$ be the mapping from inputs to discriminator features at the lth hidden layer. FMSE measures the L2 distance between $c^{(l)}(x)$ and $c^{(l)}(y)$. Notice that $c^{(l)}$ is Lipschitz continuous with constant L_l [63]. Thus $\rho_3(x,y) \le L_l\epsilon$. □

4.3 Privacy under a Public Generator

We now examine if $\mathcal{M}(\cdot,\cdot)$ allows reconstruction of authentic data. First, Proposition 2 shows that for small enough D learning rate and for D with two private layers, (ϵ,δ)-reconstruction hardness can be achieved.

Proposition 2. *Given any $\Delta\theta_{G_t}$ attained by $\mathcal{M}(\cdot,\cdot)$ for some $\theta_{D_{t-1}} \in \mathcal{H}_D$ and $x^* \in \mathcal{X}$, consider D with two private linear layers and a public sigmoid output: $D_t(x) = \sigma(a_t^T B_t x)$, where $a_t \in \mathbb{R}^l$ and $B_t \in \mathbb{R}^{l\times d_x}$. (ϵ,δ)-reconstruction hardness can be achieved if $\alpha_t \le c\epsilon^{-1}\delta^{1/d_x}$, where the constant c is problem independent.*

Proof. First, recall that the generator update follows:

$$\Delta\theta_{G_t} = \beta_t(1 - D_t(\tilde{x}))^{-1}\nabla_x D_t(\tilde{x})\nabla_\theta G_{t-1}(\tilde{z}), \tag{8}$$

where $\tilde{x} = G_{t-1}(\tilde{z})$ is a generated data point and $z \sim p_z$. Recall that for both P1 and P2, the vector $(1 - D_t(\tilde{x}))^{-1}\nabla_x D_t(\tilde{x})$ is public to the servers. We now consider the worst case where $D_t(\tilde{x})$ and $\nabla_x D_t(\tilde{x})$ are known to the attacker. This is true for P2. For P1, this is also reasonable for a successful training as $D_t(\tilde{x})$ approaches 0.5.

Given $D_t(x) = \sigma(a_t^T B_t x)$, we have

$$\nabla_x D_t(\tilde{x}) = -D_t(\tilde{x})(1 - D_t(\tilde{x}))a_t^T B_t. \tag{9}$$

Hence, $w_t^T := a_t^T B_t$ can be computed by the attacker given $\Delta\theta_{G_t}$ for any $t \in [T]$.

Since $\mathcal{M}(\cdot,\cdot)$ assumes that the update of D is performed based on a single authentic data point x^*, we have the following updates from a_{t-1} and B_{t-1}:

$$\begin{aligned} \Delta a &= \alpha_{t-1}D_{t-1}(x^*)(1 - D_{t-1}(x^*))\nabla_a a_{t-1}^T B_{t-1}x^* \\ &= \alpha_{t-1}D_{t-1}(x^*)(1 - D_{t-1}(x^*))B_{t-1}x^*, \end{aligned} \tag{10}$$

and

$$\begin{aligned} \Delta B &= \alpha_{t-1}D_{t-1}(x^*)(1 - D_{t-1}(x^*))\nabla_B a_{t-1}^T B_{t-1}x^* \\ &= \alpha_{t-1}D_{t-1}(x^*)(1 - D_{t-1}(x^*))a_{t-1}(x^*)^T \end{aligned} \tag{11}$$

Let $\Delta w = w_t - w_{t-1}$, then

$$\begin{aligned}\Delta w^T =& (a_{t-1} + \Delta a)^T (B_{t-1} + \Delta B) - a_{t-1}^T B_{t-1} \\ =& \Delta a^T B_{t-1} + a_{t-1}^T \Delta B + \Delta a^T \Delta B \\ =& (x^*)^T \left(c\|a_{t-1}\|_2^2 + c^2 w_{t-1}^T x^* + c B_{t-1}^T B_{t-1} \right),\end{aligned} \tag{12}$$

where $c = \alpha_{t-1} D_{t-1}(x^*)(1 - D_{t-1}(x^*))$ is known.

For brevity, we will omit time dependence in the following discussion. Let $g = \Delta w / c$, we have the following system of equations with respect to unknown quantities $a \in \mathbb{R}^l$, $B \in \mathbb{R}^{l \times d_x}$, and the reconstruction $\hat{x} \in \mathbb{R}^{d_x}$, where $g \in \mathbb{R}^{d_x}$, $c \in \mathbb{R}$, $w \in \mathbb{R}^{d_x}$ are known parameters:

$$\begin{aligned} g &= \left(diag\left(\|a\|_2^2 + c(\hat{x})^T B^T a \right) + B^T B \right) \hat{x} \\ w &= B^T a. \end{aligned} \tag{13}$$

We need to show that there exists an infinite number of $\hat{x}$, for each of which there exists some (a, B) that, together with the corresponding $\hat{x}$, satisfies Eq. (13).

To do so, we propose the following synthesis process. First, let $N = Q \Lambda Q^T \in \mathbb{R}^{d_x \times d_x}$ be a symmetric matrix with randomly sampled orthogonal matrix $Q \in \mathbb{R}^{d_x \times d_x}$ and positive-definite diagonal matrix $\Lambda = diag(\lambda_1, ..., \lambda_{d_x})$. Let $\lambda = \min_i \{\lambda_i\}$. Then introduce

$$M := N\gamma := diag(\|a\|_2^2 + c(\hat{x})^T B^T a) + B^T B, \tag{14}$$

with some $\gamma > 0$. Since M is positive-definite, we can get $\hat{x} = M^{-1} g$ from Eq. (13). From Eq. (14) we also have

$$B^T B = Q diag(\lambda_i \gamma - c w^T \hat{x} - \|a\|_2^2) Q^T. \tag{15}$$

Let $\|a\|_2^2 = 2\tau \|w\|_2$ where $\tau \in (0, 0.5]$. Then we have the following constraint on γ for $B^T B$ to be positive semi-definite:

$$\min_i \sigma_i^2 := \min_i \lambda_i \gamma - c w^T \hat{x} - 2\tau \|w\|_2 \geq 0. \tag{16}$$

Solve Eq. (16) to have

$$\gamma \geq \frac{\tau \|w\|_2 + \sqrt{\tau^2 \|w\|_2^2 + \lambda c g^T N^{-1} w}}{\lambda}. \tag{17}$$

Then we can rewrite $B^T B = Q \Sigma^2 Q^T$, where $\Sigma^2 = diag(\sigma_1^2, ..., \sigma_{d_x}^2)$. Therefore, B can be constructed as $B = U \bar{\Sigma} Q^T$ with some orthogonal matrix $U \in \mathbb{R}^{l \times l}$ and $\bar{\Sigma} \in \mathbb{R}^{l \times d_x}$ with diagonal elements from those of Σ.

We still need to show that there exists $\tau \in (0, 0.5]$ such that B constructed in the above satisfies the second equation in Eq. (13): $w = a^T B$. To do so, we first note that $a^T U = w^T Q \bar{\Sigma}^\dagger$, where $\bar{\Sigma}^\dagger \in \mathbb{R}^{l \times d_x}$ is the pseudo-inverse of $\bar{\Sigma}$. Now we show that there exists τ such that the following satisfies:

$$2\tau \|w\| = \|a\|^2 = \|a^T U\|^2 = \|w^T Q \bar{\Sigma}^\dagger\|^2. \tag{18}$$

Let $y = Q^T w$ and $A_i = \lambda_i \gamma - c w^T \hat{x}$, we have

$$\begin{aligned}\|w^T Q \bar{\Sigma}^\dagger\|^2 &= y^T \Sigma^{-2} y \\ &= \sum_i (A_i - 2\tau \|w\|)^{-1} y_i^2\end{aligned} \tag{19}$$

Let $A = \min_i \{A_i\}$. For τ to exist, we need to show that the following function have a root in $[0, A/2]$:

$$f(x) = \sum_i \frac{y_i^2}{A_i - x} - x. \tag{20}$$

Note that $f(0) = \sum_i y_i^2 / A_i > 0$ and $f(A/2) \leq 2\|w\|^2 / A - A/2$. Since $A \geq 2\|w\|$, $f(A/2) \leq 0$. Since f is continuous in $[0, A/2]$, $f(x)$ has a root in $[0, A/2]$.

To summarize the synthesis process: with some arbitrary orthogonal matrices Q and U, positive diagonal matrix Λ, γ satisfying Eq. (17), and τ satisfying $f(2\tau\|w\|) = 0$, we can compute M, $\bar{\Sigma}$, and $\bar{\Sigma}^\dagger$, then $\hat{x} = M^{-1} g$, $B = U \bar{\Sigma} Q^T$, and $a = U(\bar{\Sigma}^\dagger)^T Q^T w$ satisfy Eq. (13).

Since both the transformation (Q) and scaling (Λ) can be chosen freely, the above result suggests that we can choose arbitrary $\hat{x}$ within a ball of some radius R to satisfy Eq. (13). We now derive the lower bound on R:

$$\begin{aligned}\|\hat{x}\|_2^2 &= \frac{g^T Q \Lambda^{-2} Q^T g}{\gamma^2} \\ &= \frac{(g')^T \Lambda^{-2} g'}{\gamma^2} \\ &= \frac{\sum_{i=1}^{d_x} (g_i')^2 \lambda_i^{-2}}{\gamma^2} \\ &\leq \frac{\sum_{i=1}^{d_x} (g_i')^2 \lambda_i^{-2} \lambda^2}{\left(\|w\|_2 + \sqrt{\|w\|_2^2 + \lambda c^2 \sum_i (g_i')^2 \lambda_i^{-1}} \right)^2}.\end{aligned} \tag{21}$$

The lower bound of the RHS of Eq. (21) is

$$\text{RHS} \geq \frac{\sum_{i=1}^{d_x} (g_i')^2}{(1 + \sqrt{2})^2 \|w\|_2^2} = c^{-2} (1 + \sqrt{2})^{-2} \geq \frac{16}{(1 + \sqrt{2})^2} \alpha_t^{-2}. \tag{22}$$

Let $B_r(x) \subset \mathbb{R}^{d_x}$ be a ball centered at x with radius r. Then $\hat{x}$ can be arbitrarily chosen in $B_R(0)$ with radius $R = \frac{4}{\alpha_t (1 + \sqrt{2})}$.

Lastly, let $\hat{x} \in B_\epsilon(x^*)$ be successful reconstructions of x^*, and assume that $\hat{x}$ is uniformly chosen from $B_R(0)$, then to achieve (ϵ, δ)-reconstruction hardness, we need

$$\begin{aligned}& \Pr(\hat{x} \notin B_\epsilon(x^*)) = 1 - \left(\frac{\epsilon}{R} \right)^{d_x} \geq 1 - \delta \\ \Rightarrow \quad & \alpha_t \leq \frac{4}{(1 + \sqrt{2}) \epsilon} \delta^{1/d_x}.\end{aligned} \tag{23}$$

□

Remarks. Since d_x is usually large in real-world applications, reconstruction hardness can be achieved almost surely provided that $\alpha_t \leq c\epsilon^{-1}$. The proposition reveals a reasonable tradeoff: For a stronger hardness definition (larger ϵ), the training of D has to be slower. Since more private layers will further increase the difficulty of reconstruction, we conjecture that hardness is preserved for deeper private discriminators. The above result justifies the introduction of public Gs in secure GAN training, which is critical for training cost reduction due to the large sizes of generators. We reiterate that our analysis is different from PAC privacy [64]. In our setting, the attacker only observes a single gradient update $\Delta\theta_G$ within a training iteration as is specified by the protocol. PAC privacy assumes that the attacker knows the output distribution of $\mathcal{M}$.

4.4 Privacy under a Partially Private Discriminator

Now we study hardness of reconstructing x^* based on public outputs from a discriminator where only the first few layers are private. Specifically, we consider fully-connected ReLU networks $D : \mathbb{R}^{d_x} \to [0,1]$ of the following form:

$$c^{(l)} = W^{(l)}a^{(l-1)} + b^{(l)},\ \forall l = 1, ..., L,$$
$$a^{(l)} = ReLU(c^{(l)}),\ \forall l = 1, ..., L-1,$$
$$a^{(0)} = x \in \mathbb{R}^{d_x}, \quad D(x) = c^{(L)},$$

where $W^{(l)} \in \mathbb{R}^{d_l \times d_{l-1}}$ and $b^{(l)} \in \mathbb{R}^{d_l}$ are the weights and biases of layer l. $d_L = 1$. We use $\theta_D^{(l)} := (W^{(l)}, b^{(l)})$ for $l \in [L]$ to denote layer parameters. Let the first M layers be private and the rest public, i.e., $c^{(l)}$ is public only for $l \geq M$. The attack follows Prop. 3 [20] (details in Appendix A.2):

Proposition 3. *If D and its gradient updates $\nabla_{\theta_D} L$ are public, the data x^* used to compute the gradient can be reconstructed.*

Proof. Recall that the discriminator update follows:

$$\theta_{D_t} - \theta_{D_{t-1}} = -\alpha_t \nabla_{\theta_D} L(x^*), \tag{24}$$

where $L(\cdot)$ is the discriminator training objective. Using chain rule, we can get

$$\Delta W^{(1)} = -\alpha_t L_D(x^*)\delta^{(1)}(x^*)^T$$
$$\Delta b^{(1)} = -\alpha_t L_D(x^*)\delta^{(1)},$$

where

$$\delta^{(l)} = (W^{(l+1)})^T \delta^{(l+1)} \odot ReLU'(z^{(l)})$$
$$\delta^{(L-1)} = (W^{(L)})^T \odot ReLU'(z^{(L-1)}),$$

$L_D(x^*) := \nabla_D L(x^*)$, $\odot$ is element-wise product and $ReLU'(x) = 1$ if $x \geq 0$ and 0 otherwise. It is easy to see that if $\Delta\theta_D$ is known, x^* can be derived exactly from $\Delta W^{(1)}$ and $\Delta b^{(1)}$. □

Since D is partially private in our case, we study a two-step attack where the attacker first reverse engineer the private layers of D using public outputs $c^{(M)}$ based on artificial inputs, and then uses the approximation $\hat{D}$ along with $\nabla_{\theta_D} L$ to reconstruct x^*. Reconstruction hardness comes from existing studies regarding the first step. Specifically, Prop. 4 (Theorem 1.2 of [11]) states that for a fully-connected ReLU D, a function approximation algorithm exists where its data and computational complexities grow exponentially with respect to model size:

Proposition 4. *Let $x \sim \mathcal{N}(0, I)$, $S = \sum_{l=1}^{M} d_l(d_{l-1}+1)$, and $c^{(M)}(x)$ be a size-S ReLU network with depth M, Lipschitz constant at most Λ, rank of $W^{(1)}$ k, and the spectral norm of $W^{(l)}$ for $l \in [M]$ at most B. There is an algorithm that draws $d_x \log(1/\delta) \exp(poly(k, S, \Lambda/\varepsilon))B^{O(Mk)}$ samples, runs in time $\tilde{O}(d^2 \log(1/\delta)) \exp(poly(k, S, \Lambda/\varepsilon))B^{O(MkS^2)}$, and outputs a ReLU network $\hat{c}^{(M)}$ such that $\mathbb{E}[(c^{(M)}(x) - \hat{c}^{(M)}(x))^2] \leq \varepsilon$ with probability at least $1 - \delta$.*

Remarks. Note that even for spectral normalized architectures ($B \leq 1$), the exponential complexity with respect to model size S (and thus depth M) still holds. More recent studies showed that for $M = 2$, polynomial approximation algorithms exist [9, 12]. Yet for $M = 3$, Prop. 5 (Theorem 4.1 in [10]) showed that polynomial approximation cannot be achieved. Informally, the proposition states that if a 3-layer ReLU network can be learned in polynomial time, then the Learning With Rounding (LWR) problem would be solved in polynomial time. A contradiction is reached assuming hardness of LWR. We conjecture that this approximation hardness result holds for $M > 3$.

Proposition 5. *Let n be the security parameter, and fix moduli $p, q \geq 1$ such that $p, q = poly(n)$ and $p/q = poly(n)$. Let $d = n$. Let $c > 0, m = m(d) = \log^c(d)$ and $d' = d^m$. Suppose there exists a $poly(d')$-time algorithm capable of learning $poly(d')$-sized depth-2 ($M = 3$) ReLU networks under $N(0, Id_{d'})$ up to squared loss $1/poly(d')$. Then there exists a $poly(d') = 2^{\Theta(\log^{1+c} n)}$ time algorithm for $LWR_{n,p,q}$*

For non-ReLU activation functions, we use Prop. 6 (Theorem 15 in [1]), which states that the the number of samples needed for learning a general neural network is at least exponential in M. Formally,

Proposition 6. *Fix any nonlinear activation σ with the coefficient of non-linearity μ that satisfies the sub-gaussianity assumption. Let f_W be M-layer neural network with width $m = \Omega(M\frac{\mu^2}{\delta^2})$ taking inputs of dimension d with weights randomly initialized to standard Gaussians. Any algorithm that makes at most $p(d, M)$ statistical queries with tolerance $1/poly(d, M)$ and outputs a function that is $1/poly(d, M)$-correlated with $sgn(f_W)$ must satisfy $p(d, M) \geq exp(\Omega(M))$.*

Lastly, we note that the above results are concerned about approximation hardness. Yet parameter recovery, as is needed for the first attack step, is a stronger requirement, and has not been achieved through polynomial solutions even for $M = 2$ [3].

4.5 Relation between Our P2 and Federated Learning

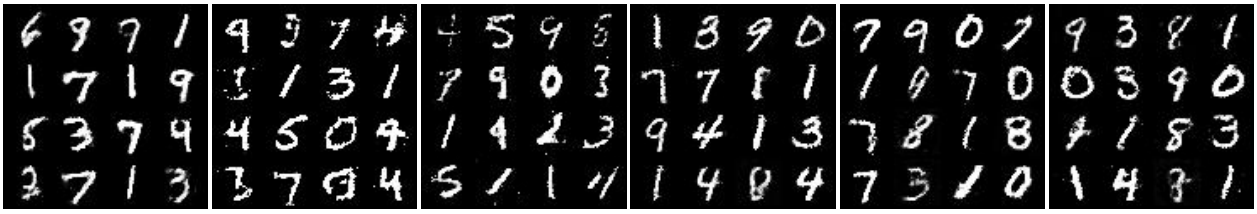
In P2, we assume servers have access to partial information of the training discriminator. We note that this mechanism of publicizing the model instead of the data is not new. In Federated Learning, the parties participating in the training process send the whole models to the central server, meaning the whole model are being made public to the servers. In our case, the servers know a part of the model. Thus, the information being made public are less in P2 compared to federated training, yet the type of information being shared is quite similar. We further quantify this difference in Section 5.3.

5 Experiment

We answer the following questions through experiments:

(1) Can P1 and P2 provide acceptable generation quality for image generation task? If so, how much faster are their learning than P0? (Sec. 5.1)
(2) How much performance boost do the proposed protocols bring to PPGAN ? (Sec. 5.1)
(3) Can P1 and P2 provide acceptable *conditional* generation quality for downstream task? (Sec. 5.2)
(4) How does P2 fare against Federated Learning and Differential Privacy in terms of reconstruction hardness? (Sec. 5.3)

To answer the first two questions, we did two experiments: one on FCGAN model on MNIST dataset, and one on DCGAN model



(a) standard GAN (b) P0 (c) P1 (d) P2 3-layer (e) P2 2-layer (f) P2 1-layer

Figure 5: Generated images from models trained on MNIST.

(a) standard (b) P1 (c) P2 3-layer (d) P2 2-layer (e) P2 1-layer

Figure 6: Generated images from models trained on CelebA with image size 64×64.

on CelebA dataset with image size 64×64. Detail of the models' architecture can be found in Appendix A.3.

To answer the third question, we conduct another experiment on conditional generation task on MNIST dataset. After training the conditional PPGAN, we generate 60000 images with similar class distributions as the MNIST training dataset and test if a classifier can be trained well on the generated dataset.

To answer the last question, we conduct a practical attack on both Federated Learning model and PPGAN to see if the set up in Protocol 2 can provide better protection.

Experiment settings. In all GAN training, we use 50k training iterations, a batch size of 32, and SGD with a learning rate of 0.1 for MNIST and 0.002 for CelebA [1]. Due to the overhead cost of secret-sharing the data, we do not use data augmentation. All data values are normalized to $[-1, 1]$ before training. We train 5 unconditional models respectively using standard GAN, P0, P2, and P2 with 1 to 3 private layers, and we train 3 conditional models respectively using standard GAN, P1, and P2 with 3 private layers. We run all experiment on local machine with 11th Gen Intel(R) Core(TM) i9-11900KF Processor with an all-core CPU frequency of 3.50GHz, 16 vCPU, 32GB RAM.

5.1 Uncoditional PPGAN on MNIST and CelebA

Generation quality and cost on MNIST. We train an FCGAN [23] on MNIST and report generation quality in Fig. 5) and Tab. 1. P1 and P2 achieve significantly lower training costs than P0: The wall-clock time is 68k seconds (19 hours) for P1, 28k seconds (8 hours) for P2 3-layer, and 11k seconds (3 hours) for P2 1-layer. Although we follow the common practice of GAN training to run the experiment only once, we note that the runtime performance is (1) not affected by the randomness of the experiments rooted from sampling, and (2) measured over 50000 iterations. Thus, the result indicates an average per iteration runtime that is reliable enough to show P1 and P2 are more efficient than the baseline.

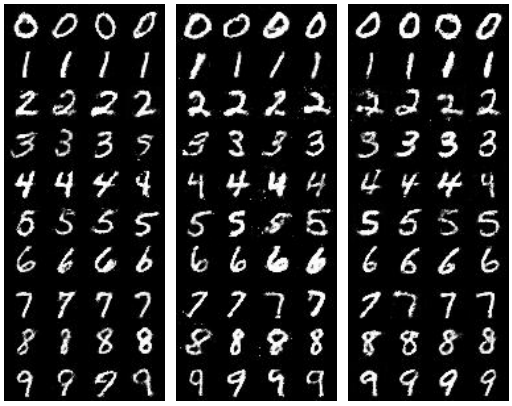
[1] Crypten only supports SGD currently. We leave MPC implementation of more advanced training algorithms for future studies.

Table 1: Runtime and generation quality comparison on MNIST. Speed-up baseline is P0.

Model	FID	Training Time (s)	Speed up
standard	115	429	235×
P0	124	101000	1.0×
P1	118	68306	1.48×
P2 3-layer	114	28289	3.57×
P2 2-layer	115	20588	4.91×
P2 1-layer	114	10580	9.55×

Table 2: Runtime and generation quality comparison on CelebA.DP,ϵ: ϵ-DP .P2,M: M-layer

Model	FID	Training Time (s)	FID<100 #Iter	FID<100 time (s)
standard	48.52	1214	12400	301.07
DP,1.39e8	445	4920	-	-
P0	-	238766	-	-
P1	94.91	119383	43200	103146
P2,3	91.34	59348	43200	51276
P2,2	98.86	41607	25200	21001
P2,1	78.84	14002	29800	8345



(a) standard GAN (b) P1 (c) P2 3-layer

Figure 7: CGAN on standard training, Protocol 1, and Protocol 2 3-layer.

Generation quality and cost on CelebA. We train a DCGAN [52] on CelebA. Fig. 6 compares generation quality of models from all protocols except P0. Since P0 uses full MPC, it fails to converge due to (i) information loss in secret sharing of data and model weights, and (ii) the intrinsic lack of convergence of SGD on GAN [7]. The wall-clock runtime and FID scores are reported in Tab. 2. The comparison shows that generation quality of P1 and P2 are visually comparable to the standard training. However, P1 and P2 still have a significant quality and training efficiency gap from the standard GAN, as is reflected through FID and the minimum number of iterations to achieve FID<100. Tab. 2 also shows that having a smaller number of private layers reduces the training cost: P1 reduces the cost by 2× from P0 and P2 by up to 16×.

5.2 Classification Performance on Conditionally Generated MNIST

We examine the generation quality of the proposed protocols with respect to downstream classification tasks. For each protocol, we generate a synthetic MNIST dataset of 60k data points from its trained model. The datasets all consist with the same label distribution as original MNIST training dataset. The qualitative visualization of our protocol and a standard CGAN training are shown in Figure 7.

For each synthetic dataset, we train a Logistic Regression (LR) and a Multi-layer Perceptron (MLP) model. We report the Area Under ROC curve (AuROC) for the resultant classifiers on the standard MNIST 10k test dataset. We compare our protocol with GW-GAN which has DP guarantee [8]. The results in Tab. 3 show that the accuracy of models trained on our synthetic dataset outperform that on [8], while being marginally lower than those trained on the authentic MNIST training set. In addition, our models achieve better Inception score yet worse FID than [8]. Note, however, that FID is defined based on the Inception model trained on ImageNet, which is not representative for MNIST.

Table 3: Comparison on classification accuracy (AuROC), Inception Score (IS) and FID Score. bold: best of secure protocol

Data	AuROC		IS	FID
	LR	MLP		
real	98.87%	99.62%	-	-
GS-WGAN	94.03%	94.74%	9.23	**61.34**
P1	96.82%	97.85%	9.49	114
P2 (3-layer)	**97.00**%	**98.12**%	**9.58**	118

Table 4: Reconstruction quality. CW-SSIM (CW) and PSNR: low value = high privacy; FMSE: high value = high privacy

Dataset	Model	CW	PSNR	FMSE
CelebA	FL GAN	0.79	32.89	7.06E-07
	DP (ϵ = 9.6)	0.27	17.94	3.77
	GP (p = 0.99)	0.43	15.30	2.77E-07
	1-layer	0.359	11.11	3.34E-4
	2-layer	0.20	**6.24**	**5.13**
	3-layer	**0.19**	6.76	2.36
MNIST	FL GAN	0.99	36.19	4.45E-05
	DP (ϵ = 9.6)	0.359	18.53	1.02E-05
	GP (p = 0.99)	0.78	26.01	0.0026
	1-layer	0.54	13.24	7.79E-06
	2-layer	**0.17**	13.11	**2.43E-04**
	3-layer	0.22	**12.33**	1.91E-04

5.3 Reconstruction Hardness

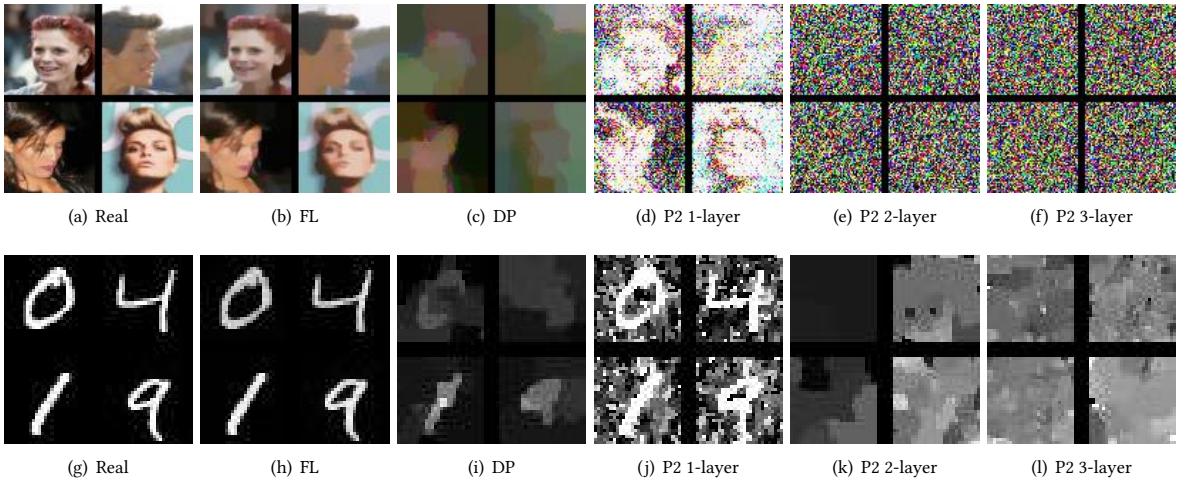
We follow the method in Sec. 4.4 (and [20]) to test the hardness of P2, FL, DP, and Gradient Pruning (GP) [39].

Attack settings: The attacker is allowed to use a varying batch size of 16 to 1024 artificial data points to reconstruct the private layers of D, and only needs to reconstruct a single authentic data point based on its output $c^{(M)}$. These parameters are chosen in favor of the attacker, while in practice a larger batch-size for the authentic data will make the reconstruction harder [20]. Note that the parameters setup is still consistent with our theoretical analysis provided in Section 4, as batch size of real data, i.e. the real input that the adversary wants to reconstruct, is always set to 1.

Protocols: We consider P2 with the first 1 to 3 layers being private. For DP, we use $\epsilon = 9.6$ and $\delta = 10^{-5}$ as in [62]. We follow the setup in [65] where we clip the gradient by $C = 0.01$ and add the noise of $\mathcal{N}(\sigma^2 C^2)$ where $\sigma = 2\frac{32}{\text{dataset-size}}\sqrt{50000\log(\frac{1}{\delta})}\Big/\epsilon$. Note that GS-WGAN [13] uses $\epsilon = 10.0$ which is easier to attack than our setting.

Metrics: We measure the similarity (or difference) between the authentic and the reconstructed images using CW-SSIM, PSNR, and Feat-MSE.

Result: Fig. 8 compares MNIST and CelebA reconstruction results for all training protocols. Quantitative results are summarized in Tab. 5. The result is consistent with our analysis: reconstruction becomes hard when the first 3 layers of D are private. We note that 2 private layers already achieved empirical hardness because while the SOTA approximation attack on 2-layer fully-connected ReLU network enjoys a polynomial data complexity [12], the amount of



(a) Real (b) FL (c) DP (d) P2 1-layer (e) P2 2-layer (f) P2 3-layer

(g) Real (h) FL (i) DP (j) P2 1-layer (k) P2 2-layer (l) P2 3-layer

Figure 8: Reconstructed images during different training protocols. Left to right: Original training data, Federated Learning, Differential Privacy (ϵ = 9.6), Protocol 2 1,2,3-layer.

Table 5: Reconstruction quality of diffusion model on MNIST dataset. CW-SSIM (CW) and PSNR: low value = high privacy; FMSE: high value = high privacy

Dataset	Model	CW	PSNR	FMSE
MNIST	FL Diffusion	0.65	19.26	0.21
	GP (p = 0.99)	0.65	19.03	0.21
	1-layer	0.38	13.34	0.21
	2-layer	**0.16**	13.06	**0.20**
	3-layer	0.21	**12.47**	0.20

data needed to launch such an attack is still much larger than the usual batch size allowed for discriminator update. Since the batch size is controlled by the training protocols, using 2 private layers in D is practically sufficient. Lastly, we note that while DP achieves some level of reconstruction hardness, the resultant DP-GAN training cannot successfully converge due to the gradient noise added in DP, leading to poor generation quality even in MNIST [62].

5.4 Reconstruction Hardness Beyond GAN model

Since our protocol P2 partially shares first layers of the network does not technically limit to GAN training, it is possible to extend the technique to other image generation model. We proposed one of such extensionn in Appendix B In this section, we want to test whether such an extension is secure given the same attack setting as the previous section.

6 Conclusion

In this paper, we presented two training protocols utilizing MPC in both unconditional and conditional GAN training. In addition, we introduced the notion of reconstruction hardness, as well as proofs and empirical results demonstrating how these protocols safeguard sensitive user data from potential reconstruction attacks by untrusted servers used for training on these data. In term of effectiveness, the proposed protocols improve the training cost by $2-16\times$ compared with the full MPC training. Plus, we show a significant improvement on conditional generation that boost the accuracy of model trained on generated data by 2-3%. Since our method achieves reconstruction hardness under attacks during mini-max training, it can be directly applied to minimization problems, e.g., training of diffusion models (see Appendix B).

Limitations. Despite the positives, our protocol still has limitations. First, our protocol only provides protection against the proposed reconstruction hardness, and provide no guarantees against other types of attack such as Membership Inference Attack. Second, our analysis rely mainly on the L2 distance, and justified this choice based on the fact that commonly used distance metrics between images (CW-SSIM, PSNR, and FMSE) are contained within large enough spheres defined by an L2 distance. This raises a limitation on defense against partial information reconstruction attack: even though we can protect partial information using a large enough L2 sphere, as long as the radius is bounded, it may provide an overly conservative upper bound on the learning rate due to the conservative translation.

Future work. We will assess potential data-efficient model approximation attacks, e.g., [12] in future work.

Acknowledgments

This work was in part supported by NSF awards #2101052, #2115075, and ARPA-H SP4701-23-C-0074. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the NSF or the U.S. Government.

References

[1] Naman Agarwal, Pranjal Awasthi, and Satyen Kale. 2020. A Deep Conditioning Treatment of Neural Networks.
[2] Sean Augenstein, H. Brendan McMahan, Daniel Ramage, Swaroop Ramaswamy, Peter Kairouz, Mingqing Chen, Rajiv Mathews, and Blaise Aguera y Arcas. 2020.

Generative Models for Effective ML on Private, Decentralized Datasets. In *International Conference on Learning Representations*. https://openreview.net/forum?id=SJgaRA4FPH

[3] Pranjal Awasthi, Alex Tang, and Aravindan Vijayaraghavan. 2021. Efficient algorithms for learning depth-2 neural networks with general relu activations. *Advances in Neural Information Processing Systems* 34 (2021), 13485–13496.

[4] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. 2017. Practical Secure Aggregation for Privacy-Preserving Machine Learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (Dallas, Texas, USA) *(CCS '17)*. Association for Computing Machinery, New York, NY, USA, 1175–1191. https://doi.org/10.1145/3133956.3133982

[5] Nicolas Carlini, Jamie Hayes, Milad Nasr, Matthew Jagielski, Vikash Sehwag, Florian Tramer, Borja Balle, Daphne Ippolito, and Eric Wallace. 2023. Extracting training data from diffusion models. In *32nd USENIX Security Symposium (USENIX Security 23)*. 5253–5270.

[6] Qi Chang, Hui Qu, Yikai Zhang, Mert Sabuncu, Chao Chen, Tong Zhang, and Dimitris Metaxas. 2020. Synthetic Learning: Learn From Distributed Asynchronized Discriminator GAN Without Sharing Medical Image Data. 13853–13863. https://doi.org/10.1109/CVPR42600.2020.01387

[7] Tatjana Chavdarova, Gauthier Gidel, François Fleuret, and Simon Lacoste-Julien. 2020. Reducing Noise in GAN Training with Variance Reduced Extragradient. arXiv:1904.08598 [stat.ML]

[8] Dingfan Chen, Tribhuvanesh Orekondy, and Mario Fritz. 2020. GS-WGAN: A Gradient-Sanitized Approach for Learning Differentially Private Generators. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., 12673–12684. https://proceedings.neurips.cc/paper_files/paper/2020/file/9547ad6b651e2087bac67651aa92cd0d-Paper.pdf

[9] Sitan Chen, Zehao Dou, Surbhi Goel, Adam Klivans, and Raghu Meka. 2023. Learning narrow one-hidden-layer relu networks. In *The Thirty Sixth Annual Conference on Learning Theory*. PMLR, 5580–5614.

[10] Sitan Chen, Aravind Gollakota, Adam Klivans, and Raghu Meka. 2022. Hardness of Noise-Free Learning for Two-Hidden-Layer Neural Networks. In *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (Eds.), Vol. 35. Curran Associates, Inc., 10709–10724. https://proceedings.neurips.cc/paper_files/paper/2022/file/45a7ca247462d9e465ee88c8a302ca70-Paper-Conference.pdf

[11] Sitan Chen, Adam R Klivans, and Raghu Meka. 2022. Learning deep relu networks is fixed-parameter tractable. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 696–707.

[12] Sitan Chen and Shyam Narayanan. 2023. A faster and simpler algorithm for learning shallow networks. *arXiv preprint arXiv:2307.12496* (2023).

[13] Xiaojun Chen, Shu Yang, Li Shen, and Xuanrong Pang. 2020. A Distributed Training Algorithm of Generative Adversarial Networks with Quantized Gradients. arXiv:2010.13359 [cs.LG]

[14] Daniel Demmler, Thomas Schneider, and Michael Zohner. 2015. ABY - A Framework for Efficient Mixed-Protocol Secure Two-Party Computation. In *22nd Annual Network and Distributed System Security Symposium, NDSS 2015, San Diego, California, USA, February 8-11, 2015*. The Internet Society.

[15] Abdulrahman Diaa, Lucas Fenaux, Thomas Humphries, Marian Dietz, Faezeh Ebrahimianghazani, Bailey Kacsmar, Xinda Li, Nils Lukas, Rasoul Akhavan Mahdavi, Simon Oya, Ehsan Amjadian, and Florian Kerschbaum. 2024. Fast and Private Inference of Deep Neural Networks by Co-designing Activation Functions. arXiv:2306.08538 [cs.CR]

[16] Cynthia Dwork and Aaron Roth. 2014. The Algorithmic Foundations of Differential Privacy. *Found. Trends Theor. Comput. Sci.* 9, 3–4 (aug 2014), 211–407. https://doi.org/10.1561/0400000042

[17] Radhika Garg, Kang Yang, Jonathan Katz, and Xiao Wang. 2023. Scalable Mixed-Mode MPC. Cryptology ePrint Archive, Paper 2023/1700. https://eprint.iacr.org/2023/1700 https://eprint.iacr.org/2023/1700.

[18] R. Garg, K. Yang, J. Katz, and X. Wang. 2024. Scalable Mixed-Mode MPC. In *2024 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, Los Alamitos, CA, USA, 109–109. https://doi.org/10.1109/SP54263.2024.00106

[19] Jonas Geiping. 2020. Inverting Gradients - How easy is it to break Privacy in Federated Learning? https://github.com/JonasGeiping/invertinggradients.

[20] Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. 2020. Inverting gradients-how easy is it to break privacy in federated learning? *Advances in Neural Information Processing Systems* 33 (2020), 16937–16947.

[21] Garrett B. Goh, Nathan O. Hodas, and Abhinav Vishnu. 2017. Deep Learning for Computational Chemistry. arXiv:1701.04503 [stat.ML]

[22] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. 1986. How to Construct Random Functions. *J. ACM* 33, 4 (aug 1986), 792–807. https://doi.org/10.1145/6490.6503

[23] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger (Eds.). Curran Associates, Inc., 2672–2680. http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf

[24] 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.

[25] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. 2017. Improved training of wasserstein gans. *Advances in neural information processing systems* 30 (2017).

[26] Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising Diffusion Probabilistic Models. arXiv:2006.11239 [cs.LG]

[27] Alain Horé and Djemel Ziou. 2010. Image Quality Metrics: PSNR vs. SSIM. In *2010 20th International Conference on Pattern Recognition*. 2366–2369. https://doi.org/10.1109/ICPR.2010.579

[28] Xiaoyang Hou, Jian Liu, Jingyu Li, Yuhan Li, Wen jie Lu, Cheng Hong, and Kui Ren. 2023. CipherGPT: Secure Two-Party GPT Inference. Cryptology ePrint Archive, Paper 2023/1147. https://eprint.iacr.org/2023/1147 https://eprint.iacr.org/2023/1147.

[29] N. Jawalkar, K. Gupta, A. Basu, N. Chandran, D. Gupta, and R. Sharma. 2024. Orca: FSS-based Secure Training and Inference with GPUs. In *2024 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, Los Alamitos, CA, USA, 66–66. https://doi.org/10.1109/SP54263.2024.00063

[30] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. 2020. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 8110–8119.

[31] B. Knott, S. Venkataraman, A.Y. Hannun, S. Sengupta, M. Ibrahim, and L.J.P. van der Maaten. 2021. CrypTen: Secure Multi-Party Computation Meets Machine Learning. In *arXiv 2109.00984*.

[32] Brian Knott, Shobha Venkataraman, Awni Hannun, Shubho Sengupta, Mark Ibrahim, and Laurens van der Maaten. 2021. Crypten: Secure multi-party computation meets machine learning. *Advances in Neural Information Processing Systems* 34 (2021).

[33] Marek Kowalski, Stephan J Garbin, Virginia Estellers, Tadas Baltrušaitis, Matthew Johnson, and Jamie Shotton. 2020. Config: Controllable neural face image generation. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XI 16*. Springer, 299–315.

[34] Lalit Kumar and Dushyant Kumar Singh. 2023. A comprehensive survey on generative adversarial networks used for synthesizing multimedia content. *Multimedia Tools Appl.* 82, 26 (mar 2023), 40585–40624. https://doi.org/10.1007/s11042-023-15138-x

[35] Lan Lan, Lei You, Zeyang Zhang, Zhiwei Fan, Weiling Zhao, Nianyin Zeng, Yidong Chen, and Xiaobo Zhou. 2020. Generative adversarial networks and its applications in biomedical informatics. *Frontiers in public health* 8 (2020), 164.

[36] Dacheng Li, Hongyi Wang, Rulin Shao, Han Guo, Eric Xing, and Hao Zhang. 2023. MPCFORMER: FAST, PERFORMANT AND PRIVATE TRANSFORMER INFERENCE WITH MPC. In *The Eleventh International Conference on Learning Representations*. https://openreview.net/forum?id=CWmvjOEhgH-

[37] Zinan Lin, Alankar Jain, Chen Wang, Giulia Fanti, and Vyas Sekar. 2020. Using gans for sharing networked time series data: Challenges, initial promise, and open questions. In *Proceedings of the ACM Internet Measurement Conference*. 464–483.

[38] Jian Liu, Mika Juuti, Yao Lu, and N. Asokan. 2017. Oblivious Neural Network Predictions via MiniONN Transformations. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (Dallas, Texas, USA) *(CCS '17)*. Association for Computing Machinery, New York, NY, USA, 619–631. https://doi.org/10.1145/3133956.3134056

[39] Sheng Liu, Zihan Wang, Yuxiao Chen, and Qi Lei. 2024. Data Reconstruction Attacks and Defenses: A Systematic Evaluation. arXiv:2402.09478 [cs.CR] https://arxiv.org/abs/2402.09478

[40] H. Brendan McMahan, Eider Moore, Daniel Ramage, and Blaise Agüera y Arcas. 2016. Federated Learning of Deep Networks using Model Averaging. *CoRR* abs/1602.05629 (2016). arXiv:1602.05629 http://arxiv.org/abs/1602.05629

[41] Mehdi Mirza and Simon Osindero. 2014. Conditional Generative Adversarial Nets. *CoRR* abs/1411.1784 (2014). arXiv:1411.1784 http://arxiv.org/abs/1411.1784

[42] Payman Mohassel and Peter Rindal. 2018. ABY3: A Mixed Protocol Framework for Machine Learning. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security* (Toronto, Canada) *(CCS '18)*. Association for Computing Machinery, New York, NY, USA, 35–52. https://doi.org/10.1145/3243734.3243760

[43] Payman Mohassel, Mike Rosulek, and Ni Trieu. 2020. Practical Privacy-Preserving K-means Clustering. *Proceedings on Privacy Enhancing Technologies (PETS)* 2020, 4 (2020), 414–433.

[44] Payman Mohassel and Yupeng Zhang. 2017. Secureml: A system for scalable privacy-preserving machine learning. In *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 19–38.

[45] Payman Mohassel and Yupeng Zhang. 2017. SecureML: A System for Scalable Privacy-Preserving Machine Learning. In *2017 IEEE Symposium on Security and Privacy (SP)*. 19–38. https://doi.org/10.1109/SP.2017.12

[46] Fania Mokhayeri, Kaveh Kamali, and Eric Granger. 2020. Cross-domain face synthesis using a controllable GAN. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*. 252–260.
[47] Alhassan Mumuni and Fuseini Mumuni. 2022. Data augmentation: A comprehensive survey of modern approaches. *Array* 16 (2022), 100258. https://doi.org/10.1016/j.array.2022.100258
[48] Mukrin Nakhwan and Rakkrit Duangsoithong. 2022. Comparison Analysis of Data Augmentation using Bootstrap, GANs and Autoencoder. In *2022 14th International Conference on Knowledge and Smart Technology (KST)*. 18–23. https://doi.org/10.1109/KST53302.2022.9729065
[49] Qi Pang, Yuanyuan Yuan, and Shuai Wang. 2024. MPCDiff: Testing and Repairing MPC-Hardened Deep Learning Models. https://doi.org/10.14722/ndss.2024.23380
[50] Qi Pang, Jinhao Zhu, Helen Möllering, Wenting Zheng, and Thomas Schneider. 2023. BOLT: Privacy-Preserving, Accurate and Efficient Inference for Transformers. Cryptology ePrint Archive, Paper 2023/1893. https://eprint.iacr.org/2023/1893 https://eprint.iacr.org/2023/1893.
[51] Rahul Rachuri and Ajith Suresh. 2019. Trident: Efficient 4PC Framework for Privacy Preserving Machine Learning. *CoRR* abs/1912.02631 (2019). arXiv:1912.02631 http://arxiv.org/abs/1912.02631
[52] Alec Radford, Luke Metz, and Soumith Chintala. 2016. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. arXiv:1511.06434 [cs.LG]
[53] Mohammad Rasouli, Tao Sun, and Ram Rajagopal. 2020. FedGAN: Federated Generative Adversarial Networks for Distributed Data. *CoRR* abs/2006.07228 (2020). arXiv:2006.07228 https://arxiv.org/abs/2006.07228
[54] Deevashwer Rathee, Anwesh Bhattacharya, Divya Gupta, Rahul Sharma, and Dawn Song. 2023. Secure Floating-Point Training. In *32nd USENIX Security Symposium (USENIX Security 23)*. USENIX Association, Anaheim, CA, 6329–6346. https://www.usenix.org/conference/usenixsecurity23/presentation/rathee
[55] Deevashwer Rathee, Anwesh Bhattacharya, Divya Gupta, Rahul Sharma, and Dawn Song. 2023. Secure Floating-Point Training. Cryptology ePrint Archive, Paper 2023/467. https://eprint.iacr.org/2023/467 https://eprint.iacr.org/2023/467.
[56] M. Sadegh Riazi, Christian Weinert, Oleksandr Tkachenko, Ebrahim M. Songhori, Thomas Schneider, and Farinaz Koushanfar. 2018. Chameleon: A Hybrid Secure Computation Framework for Machine Learning Applications. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3196494.3196522
[57] Ahmed Salem, Giovanni Cherubin, David Evans, Boris Köpf, Andrew Paverd, Anshuman Suri, Shruti Tople, and Santiago Zanella-Béguelin. 2023. SoK: Let the Privacy Games Begin! A Unified Treatment of Data Inference Privacy in Machine Learning. arXiv:2212.10986 [cs.LG] https://arxiv.org/abs/2212.10986
[58] Mehul P. Sampat, Zhou Wang, Shalini Gupta, Alan Conrad Bovik, and Mia K. Markey. 2009. Complex Wavelet Structural Similarity: A New Image Similarity Index. *IEEE Transactions on Image Processing* 18, 11 (2009), 2385–2401. https://doi.org/10.1109/TIP.2009.2025923
[59] Axel Sauer, Katja Schwarz, and Andreas Geiger. 2022. Stylegan-xl: Scaling stylegan to large diverse datasets. In *ACM SIGGRAPH 2022 conference proceedings*. 1–10.
[60] Youssef Skandarani, Pierre-Marc Jodoin, and Alain Lalande. 2023. Gans for medical image synthesis: An empirical study. *Journal of Imaging* 9, 3 (2023), 69.
[61] Jiaming Song, Chenlin Meng, and Stefano Ermon. 2022. Denoising Diffusion Implicit Models. arXiv:2010.02502 [cs.LG]
[62] R. Torkzadehmahani, P. Kairouz, and B. Paten. 2019. DP-CGAN: Differentially Private Synthetic Data and Label Generation. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. IEEE Computer Society, Los Alamitos, CA, USA, 98–104. https://doi.org/10.1109/CVPRW.2019.00018
[63] Aladin Virmaux and Kevin Scaman. 2018. Lipschitz regularity of deep neural networks: analysis and efficient estimation. In *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (Eds.), Vol. 31. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2018/file/d54e99a6c03704e95e6965532dec148b-Paper.pdf
[64] Hanshen Xiao and Srinivas Devadas. 2023. PAC Privacy: Automatic Privacy Measurement and Control of Data Processing. arXiv:2210.03458 [cs.CR]
[65] Liyang Xie, Kaixiang Lin, Shu Wang, Fei Wang, and Jiayu Zhou. 2018. Differentially Private Generative Adversarial Network. *CoRR* abs/1802.06739 (2018). arXiv:1802.06739 http://arxiv.org/abs/1802.06739
[66] Andrew C Yao. 1982. Protocols for secure computations. In *23rd annual symposium on foundations of computer science (sfcs 1982)*. IEEE, 160–164.
[67] Xin Yi, Ekta Walia, and Paul Babyn. 2019. Generative adversarial network in medical imaging: A review. *Medical image analysis* 58 (2019), 101552.
[68] Xun Yuan, Yang Yang, Prosanta Gope, Aryan Pasikhani, and Biplab Sikdar. 2024. VFLGAN: Vertical Federated Learning-based Generative Adversarial Network for Vertically Partitioned Data Publication. *Proceedings on Privacy Enhancing Technologies* 2024, 4 (Oct. 2024), 840–858. https://doi.org/10.56553/popets-2024-0144
[69] Joshua C. Zhao, Atul Sharma, Ahmed Roushdy Elkordy, Yahya H. Ezzeldin, Salman Avestimehr, and Saurabh Bagchi. 2023. LOKI: Large-scale Data Reconstruction Attack against Federated Learning through Model Manipulation. arXiv:2303.12233 [cs.LG]
[70] Yabing Zhu, Yanfeng Zhang, Huili Yang, and Fangjing Wang. 2019. GANCoder: An Automatic Natural Language-to-Programming Language Translation Approach based on GAN. arXiv:1912.00609 [cs.CL]

A More details about implementation

A.1 Federated Learning

We follow the Federated Averaging algorithm [40] in our experiment in Section 5. We note that our implementation is for pure Federated Learning without further defense such as Differential Privacy (which is separately compared in Section 5) and Secure Aggregation. The details of the algorithm is shown in Algorithm 1. We left out Secure Aggregation due to the following two reasons:

- Firstly, the Inverting Gradient attack that we used works even in the presence of Secure Aggregation when the total batch size of all clients per round is small. As tested in [20], it works for up to 100 input images. Since we study worst-case scenarios with batch size of 1, adding Secure Aggregation does not help increase security in set up of our experiment.
- Secondly, as pointed out in [69], when the total batch size is large, Secure Aggregation fails if the central server of federated learning has the ability to modify the network architecture. In contrary, under such a threat model, our training protocols are still robust to inverting gradient attacks because we do not use a central server, i.e., the semi-honest servers cannot altogether collude and change the network architecture. Based on these reasons, we do not consider secure aggregation in this paper.

Algorithm 1: FederatedAveraging. The K clients are indexed by k; C is the fraction of clients joining each round, T is the number of training rounds, and n_k is the size of dataset hold by client C_k.

Server execute
initialize weight of model w_0
for each round $t = 1, \dots, T$ **do**
  $m \leftarrow \max(C \cdot K, 1)$
  $S_t \leftarrow$ (random set of m clients)
  for each client $k \in S_t$ **do**
    Client C_k train locally, return w_{t+1}^k to server
  end for
  $m_t \leftarrow \sum_{k \in S_t} n_k$
  $w_{t+1} \leftarrow \sum_{k \in S_t} \frac{n_k}{m_t} w_{t+1}^k$
end for
return w_T

A.2 Inverting gradient attack for reconstruction

Algorithm 2 describes the inverting gradient attack from [20], whose goal is to reconstruct the training data from the discriminator gradient computed based on that data.

Algorithm 2: InvertingGradient. **Parameters:** Model D with parameters θ, loss function L, model input x, real input to reconstruct x^*, label $y = 1$ since x^* is authentic. TV is the total variation.

$x \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
optimizer $\leftarrow$ LFBGS(x)
repeat
  loss $\leftarrow 1 - \frac{\langle \nabla_\theta L(x_i, y), \nabla_\theta L(x^*, y) \rangle}{\|\nabla_\theta L(x_i, y)\| \|\nabla_\theta L(x^*, y)\|} + 0.1\text{TV}(x)$
  $x_{i+1} \leftarrow$ optimizer.step(loss)
until converge

A.3 Model Architecture

Tables 6 to 9 present model architectures used in experiments.

Table 6: FCGAN Generator Architecture

Generator	Activation	Output shape
Input noise	-	1x100
FC Layer	LeakyReLU	1x64
FC Layer	-	1x128
BatchNorm1d	-	1x128
FC Layer	-	1x256
BatchNorm1d	-	1x256
FC Layer	Tanh	1x784

Table 7: FCGAN Discriminator Architecture

Discriminator	Activation	Output shape
Input image	-	1x784
FC Layer	LeakyReLU	1x512
FC Layer	LeakyReLU	1x256
FC Layer	LeakyReLU	1x128
Fully Connected	Sigmoid	1

Table 8: DCGAN Generator Architecture

Generator	Activation	Output shape
Input noise	-	1x100
Fully Connected	LeakyReLU	128x256
Upsample	-	128x32x32
Conv 3x3	LeakyReLU	128x32x32
Upsample	-	128x64x64
Conv 3x3	LeakyReLU	64x64x64
Conv 3x3	Tanh	3x64x64

Table 9: DCGAN Discriminator Architecture

Discriminator	Activation	Output shape
Input image	-	3x64x64
Conv 3x3	LeakyReLU	16x64x64
Conv 3x3	LeakyReLU	32x64x64
Conv 3x3	LeakyReLU	64x64x64
Conv 3x3	LeakyReLU	128x64x64
Fully Connected	Sigmoid	1

B Achieving Reconstruction Hardness during Diffusion Training

B.1 DDPM: Forward Process

The forward process of a diffusion model is structured as a Markov chain process that gradually injects Gaussian noise to the data according to a variance scheduler $\beta_1, ..., \beta_T$

$$\begin{aligned} q(x_t \mid x_{t-1}) &:= \mathcal{N}(x_t; \sqrt{1-\beta_t} x_{t-1}, \beta_t \mathbf{I}) \\ q(x_{1:T} \mid x_0) &:= \prod_{t=1}^{T} q(x_t \mid x_{t-1}) \end{aligned} \tag{25}$$

Where the forward process can be further derived into a closed form solution that can sample x_t at any arbitrary times steps give x_0, using the notation $\alpha := 1 - \beta_t$, and $\bar{\alpha}_t := \prod_{s=1}^{t} \alpha_s$:

$$q(x_t \mid x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t} x_0, (1 - \bar{\alpha}_t)\mathbf{I}) \tag{26}$$

B.2 DDPM: Backward Process

The reverse process follows the Markov chain assumption given learned Gaussian transitions starting at $p(x_t) = \mathcal{N}(x_t; \mathbf{0}, \mathbf{I})$:

$$\begin{aligned} p_\theta(x_{0:T}) &:= p(x_t) \prod_{t=1}^{T} p_\theta(x_{t-1} \mid x_t) \\ p_\theta(x_{t-1} \mid x_t) &:= \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)) \end{aligned} \tag{27}$$

The reverse process $q(x_t \mid x_{t-1})$ is tractable when conditioned on x_0 [26]:

$$q(x_{t-1} \mid x_t, x_0) = \mathcal{N}(x_{t-1}; \tilde{\mu}_t(x_t, x_0), \tilde{\beta}_t \mathbf{I}) \tag{28}$$

Where,

$$\begin{aligned} \tilde{\mu}_t(x_t, x_0) &:= \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t} x_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t} x_t \\ \tilde{\beta}_t &:= \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t} \beta_t \end{aligned} \tag{29}$$

By further reparameterization of eq. 26 as $x_t(x_0, \epsilon) = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon$ for $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, we can get:

$$\begin{aligned} \mu_\theta(x_t, t) &= \tilde{\mu}_t(x_t, \frac{1}{\sqrt{\bar{\alpha}_t}}(x_t - \sqrt{1-\bar{\alpha}_t}\epsilon_\theta(x_t))) \\ &= \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t, t)) \end{aligned} \tag{30}$$

B.3 DDPM: Training Procedure

The training objective can be simplified as:

$$L_{\text{simple}}(\theta) := \mathbb{E}_{t, x_0, \epsilon}[\|\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t} x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t)\|^2] \tag{31}$$

With Eq. (31), Algorithm 3 presents the complete training procedure, and Algorithm 4 presents the sampling procedure based on Markov chain assumption following Eq. (28):

B.4 Partially-private DDPM

Similar to P2 in Section 4.4, we can design a partially-private DDPM protocol, see Algorithm 5. We have two remarks at applying P2 to diffusion:

Algorithm 3: Training

repeat
  $x_0 \sim q(x_0)$
  $t \sim \text{Uniform}(\{1, \dots, T\}))$
  $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  Take gradient descent step on
    $\nabla_\theta \|\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t} x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t)\|^2$
until converged

Algorithm 4: Sampling

$x_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
for $t = T, \dots, 1$ **do**
  $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
  $x_{t-1} = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t, t)) + \sigma_t \mathbf{z}$
end for

- **Remarks 1** In the training process, the share of the random noise $[\![\epsilon]\!]$ can be locally sampled without any communication between the servers. This is owing to the fact that the sum of Gaussian distributed variables is Gaussian.
- **Remarks 2** In the training process, we need to mask information of $x_t, \forall t \in \{1, \dots, T\}$. Because $x_0 = \frac{1}{\sqrt{\bar{\alpha}_t}}\mathbb{E}(x_t)$, the adversary can learn x_0 with high probability after it gets enough samples by calculating the average values.

Algorithm 5: Partially Private DDPM Training

repeat
  The servers locally get $[\![x_0]\!]$ from the shared dataset, they sync to get the same data point by using the same predefined random seed s
  The servers locally sample the same $t \sim \text{Uniform}(\{1, \dots, T\}))$
  Each server $\mathcal{S}_i$ locally sample $[\![\epsilon]\!] := \epsilon_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I}/n)$
  n servers use MPC to calculate $\epsilon_{\theta,\text{pri}}(\sqrt{\bar{\alpha}_t} x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t)$, and make this value public.
  Each server calculate the loss and update ϵ_{pub} using gradient descent on
    $\|\epsilon - \epsilon_{\theta,\text{pub}}(\epsilon_{\theta,\text{pri}}(\sqrt{\bar{\alpha}_t} x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t))\|^2$
  n servers together use MPC to update private $\epsilon_{\theta,\text{pri}}$:
    $\nabla_{\text{pri}} \|\epsilon - \epsilon_{\theta,\text{pub}}(\epsilon_{\theta,\text{pri}}(\sqrt{\bar{\alpha}_t} x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t))\|^2$
until converged

The reconstruction hardness of this secure training is the same as that of P2.