On the Hardness of Input Reconstruction Attack via Gradient Sharing in Federated Learning: A Cryptographic View

Abstract—**Federated learning (FL) has become one of the most popular paradigms for allowing collaborative learning among multiple parties. It is presented to be privacy-friendly by guaranteeing that local data never "leaves" personal devices but only certain intermediate parameters such as gradients or other model updates are shared. There are few analytical results on the problem of reconstructing the input training samples from gradients, but they all consider very restrictive settings, for example, the batch size should be small. In this paper, we take a step further to investigate this problem more deeply from a cryptographic view. Considering a multi-layer perceptron (MLP) network with ReLU as the activation function, we can mathematically formulate the problem of input reconstruction using the gradient information shared in FL, as a classical cryptographic problem called hidden subset sum problem (HSSP). HSSP is an extension of the well-known NP-complete problem – subset sum problem (SSP) and is an established tool for cryptanalysis. Via existing HSSP tools we can analytically show that the difficulty of input reconstruction attack depends heavily on the parameter of batch size, denoted as B. In particular, the time complexity of existing HSSP solvers is approximately $\mathcal{O}(B^9)$. Via this analytical result, we further show that applying secure data aggregation techniques such as homomorphic encryption and secure multiparty computation, would be a strong defense as it directly increases the time complexity to $\mathcal{O}(N^9 B^9)$ where N is the number of local clients in FL. To the best of our knowledge, our work is the first to mathematically formulate the privacy problem in FL as a cryptographic problem such that a concrete and rigorous analysis can be conducted.**

Index Terms—**Privacy, federated learning, gradient leakage, input reconstruction attack, hidden subset sum problem, subset sum problem**

1. Introduction

Big data comes with big challenges. Nowadays user data are frequently gathered and stored in portable devices such as phones and tablets [1], [2]. Processing such big amount of data over numerous personal devices poses severe privacy concerns as the involved data often contains personal information such as location and personal profile, etc. FL [3]–[5] was introduced to address such privacy concerns by allowing multiple devices to collaboratively train a shared model while keeping the personal data on the local devices. The FL framework often consists of a number of local participants, which have the local training dataset, and a centralized server for coordination. Each participant is allowed to send intermediate model updates such as gradients learned by local training data to the server, and the server updates the global model by aggregating the local gradients together.

The claim that FL is privacy-preserving relies on the assumption that sharing intermediate model updates will not leak users' private training data. This assumption is, however, being challenged in prior work. One of the earliest works on privacy issues of FL is the DLG attack (Deep Leakage from Gradient) [6], which shows that FL systems are susceptible to gradient inversion attacks, i.e., reversing the original training data from the observed gradient information. The main idea of DLG is to optimize synthesized data such that its gradients match the real gradient produced by the real training data. Many studies further improve the DLG attack by increasing the attack efficiency, improving the resolution of the reconstructed data, or increasing the batch size [7]–[17]. Besides the optimization-based approaches, another line of work seek to analytically analyze the privacy threat in FL. Geiping et al. [8], Fowl et al [12] and Boenisch et al. [10] show that the training data can be perfectly reconstructed under certain restrictive settings, such as only one sample being activated within a mini-batch or the case that the attacker can maliciously change the model parameters or the architectures.

Despite the remarkable progress in breaching the privacy in FL, it has been argued in [18] that FL might not be vulnerable to privacy attacks because existing attacks are effective only under specific and sometimes impractical assumptions. As an example, the batch size, i.e., how many training samples are processed before updating the model, is an important parameter of the gradient inversion attack of FL. Existing attacks are evaluated with limited setups that the batch size is quite small, or require additional assumptions such that the input samples within a batch should not share the same class label. For a reasonable batch size such as 128 or 256, the attack performances deteriorate severely [9], [11].

The debate on whether FL is vulnerable to privacy attacks in practical settings has caught our attention and motivated us to investigate it further. More specifically, our research question is as follows: **Is it possible to mathematically analyze the difficulty of the input reconstruction attack caused by gradient sharing in FL? For example,**

how does the difficulty change with relevant parameters such as the batch size or the number of neurons?

To answer this research question, we consider an MLP with ReLU [19] activation function. We mathematically show that the gradients of the first layer produced by a mini-batch are equal to the product of a weight matrix and an input matrix containing the training samples in this mini-batch. The weight matrix can be further decoupled as the element-wise multiplication of two matrices: one is the partial gradient of the loss w.r.t. the output of the first layer, and the other is the binary matrix caused by the activation.With this formulation, we can further simplify it as a classical cryptographic problem called hidden subset sum problem (HSSP) [20]–[22]. Therefore, we can adopt cryptographic tools for analyzing the hardness of the input reconstruction attack in FL. In particular, the hardness of HSSP is closely related to the well-known subset sum problem (SSP), which has been proven to be an NP-complete problem [23]. We show that when increasing the batch size, reconstructing the training samples from the gradient information is more challenging, or at least computationally heavier. Moreover, via HSSP we can analytically show that the batch size is indeed a crucial parameter for the hardness of the problem. For certain parameter settings, the time complexity of existing HSSP solvers is approximately $\mathcal{O}(B^9)$, where B denotes the batch size. Hence, such theoretical results explain the heuristic observation of why existing gradient inversion attacks fail for large batch size cases. As for the number of neurons, it also affects the time complexity but does not dominate if it is of an appropriate size.

As a marriage of both cryptography and machine learning, our work has significant impacts on both communities. On one hand, machine learning models often behave like a black-box lacking reasoning and explainability. Our work calls for more machine learning researchers to reinspect the privacy of machine learning models from a cryptographic view. Thereby a concrete and rigorous analysis can be conducted, e.g., the time complexity given by the batch size in our case. On the other hand, our work inspires how variations of cryptographic building blocks and hard problems can be constructed in practice and how their general cases can be verified in machine learning.

1.1. Paper contribution

1) To the best of our knowledge, this is the first work that mathematically formulates the gradient leakage in FL as a cryptographic problem known as HSSP.
2) With the help of cryptographic tools in HSSP, we analytically show that reconstructing the input samples from gradients is directly dependent on the batch size, i.e., a time complexity of $\mathcal{O}(B^9)$. This explains why reconstructing the input training sample is very difficult for a large batch size.
3) With the analytical complexity, we show that applying secure aggregation techniques, such as secret sharing [24] or homomorphic encryption [25] to share global gradients aggregated over all participants instead of local gradients of the individual participant, is a strong defense as it would directly increase the time complexity from $\mathcal{O}(B^9)$ to $\mathcal{O}(N^9B^9)$, where N denotes the total number of participants.

1.2. Paper outline

The rest of the paper is organized as follows. Section 2 provides a brief summary of existing privacy attacks that have been developed for FL. Section 3 briefly introduces the fundamentals of FL and the corresponding HSSP, which are necessary for comprehending the problem formulation that will be presented later. Section 4 shows how to formulate the problem of reconstructing input training data using the observed gradients, and further connects it to HSSP. Section 5 introduces the fundamentals of lattices and briefly describes the existing attacks for solving HSSP. Time complexity analysis and a typical defense are discussed in Section 6. Numerical results and discussions are given in Section 7. Finally, conclusions are given in Section 8.

2. Related work

2.1. Privacy in FL

Concerning privacy in machine learning models, there are mainly three types of privacy attack including 1) membership inference attack [26]–[30] where the goal of the attacker is to determine whether a particular sample was used in training or not; 2) property inference attack [31], [32] which aims to infer certain properties of the input training data such as gender, class label etc; and 3) input reconstruction attack (e.g., model inversion attack [33], [34]) which attempts to reconstruct the original training samples. All of these attacks have also been investigated in FL. It is shown that by exploiting the model updates shared to the server, an attacker can infer sensitive information such as the training data's properties [35]–[37] and membership information [36], or even reconstruct the input training samples [6]–[17]. In this paper, we focus on the last threat, i.e., input reconstruction attack, as it is intensively investigated in the context of FL and it raises more severe privacy concerns than the other attacks.

2.2. Gradient inversion attack

Gradient inversion attack is the most popular input reconstruction attack in FL as the shared information is often the updated gradients. There are mainly two types of approaches.

Optimization based approaches: DLG [6] is the first proposed gradient inversion attack, which demonstrates that it is possible to reconstruct the training data with high fidelity using the shared gradient information. The main observation is that if two data points are similar to each other, their gradients might also be similar. DLG first initializes a random noise image called dummy data and produces a corresponding dummy gradient, it then iteratively optimizes the dummy data to resemble the training sample by minimizing the distance between the dummy gradient and the real gradient generated by the real training data. The performance of DLG is limited as the optimization is hard to converge and it requires that the image is of low resolution and the batch size should be small. To address these limitations, a rich line of work [7]–[15] try to propose more advanced gradient inversion attacks, e.g., increasing the optimization efficiency and producing accurate reconstructed data for high-resolution images [7], [8]. Yin et al. [9] further improved the optimization efficiency by adding regularization and the attack can work with the batch size up to 48. However, it requires that one batch cannot contain two images from the same class. Yang et al. [15] optimizes the attack efficiency and considers a more challenging case where the gradients are compressed.

Given a reasonable batch size such as 128 the reconstructed data samples of existing work are often of low fidelity [9], [11]. Multiple reasons might cause such poor performances. 1) The optimization-based approaches suffer from the local minima problem and thus the reconstructed samples are often quite different from the original training data [9]. 2) The fundamental assumption that minimizing the distance between gradients would make the synthesized data similar to the original training data is not necessarily true, as different mini-batch of data may produce almost identical gradients [38].

Analytical approaches There are also a few theoretical analyses on the privacy leakage caused by gradient sharing. Geiping et al. [8] theoretically prove that it is possible to perfectly reconstruct the training samples regardless of trained or untrained models. It only considers restrictive settings, e.g., only one sample is activated among a mini-bath. Fowl et al [12] showed that it is possible to reconstruct the input by assuming a stronger threat model where the central server is active, i.e., it can change the model architecture. Similarly, Boenisch et al. [10] showed that perfect input reconstruction is possible by assuming that the central server can manipulate the weights maliciously.

3. Preliminaries

In this section, we review the necessary fundamentals for the rest of the paper.

3.1. Notations

In this work, lowercase letters x, lowercase boldface letters $\mathbf{x}$ and uppercase boldface letters $\mathbf{X}$ stand for scalars, vectors and matrices, respectively. The i-th entry of the vector $\mathbf{x}$ is denoted either as x_i or $\mathbf{x}[i]$. Similarly, the (i,j)-th entry of the matrix $\mathbf{X}$ is either denoted as $x_{i,j}$ or $\mathbf{X}[i;j]$. $\mathbf{X}[i;:]$ and $\mathbf{X}[:;i]$ denote the i-th row and column of matrix $\mathbf{X}$, respectively. $[\mathbf{x}_1,\ldots,\mathbf{x}_n]$ and $[\mathbf{x}_1;\ldots;\mathbf{x}_n]$ denote the row-wise and column-wise concatenation of the corresponding vectors. $\mathbf{X}^\top$ denotes the transpose of $\mathbf{X}$. $\mathrm{diag}\{\mathbf{X}\}$ denotes taking the diagonal elements of the square matrix $\mathbf{X}$. $\odot$ denotes element-wise multiplication. $\mathbf{1}_r$ denotes the vector of all ones of size r and similarly $\mathbf{I}_r$ denotes the identity matrix of size r.

3.2. Fundamentals of machine learning

Neural networks: Let $f_{\mathbf{W}} : \mathbb{R}^m \rightarrow \{1,\cdots,k\}$ be a k-class classifier, defined as a sequence of layers parameterized by trainable weights $\mathbf{W}$. Each layer comprises a linear operation paired with a non-linear activation function such as ReLU. A deep neural network often consists of many different layers. The main objective of the model $f_{\mathbf{W}}$ is to map an input to its desired ground-truth label, denoted as $\mathbf{x}_i$ and l_i. As a result, the model weights $\mathbf{W}$ are updated through an iterative training process, e.g., the mini-batch Stochastic Gradient Descent (SGD). It repeats the following steps for modifying the initial $\mathbf{W}$: 1) sample a mini-batch of size B from the training data $\{(\mathbf{x}_j,l_j)\}_{j=1}^{B}$; 2) execute a forward pass through the model to generate predictions for this mini-batch; 3) calculate the difference between the predictions and the ground-truth labels, known as the loss $\mathcal{L}$; 4) compute the gradient of $\mathcal{L}$ w.r.t. the weights, called the weight gradient $\mathbf{G}$, and update the weights accordingly.

Federated learning: Suppose there are N different users and each obtains a local training dataset. They would like to cooperate together to jointly learn a shared model based on their datasets. Collecting the datasets from individual users centrally would be costly and privacy-conflicting as the local data contains sensitive personal information. FL allows these users to cooperate without requiring each user to share its raw data directly. Let $f_{\mathbf{W}}^{(t)}(\cdot)$ be the model with its weights $\mathbf{W}^{(t)}$ at iteration t. FL works through the following steps:

1) Initialization: at iteration $t=0$, the central server, denoted as C, randomly initializes the weights $W^{(0)}$ of the global model.
2) Local model training: at each iteration t, the server first randomly selects a subset of users and each user i receives the model $f_{\mathbf{W}}^{(t)}(\cdot)$ from the server C. Each selected user u_i then calculates the local gradient $\mathbf{G}_i^{(t)}$ for $f_{\mathbf{W}}^{(t)}(\cdot)$ based on one mini-batch sampled from their local dataset $\{(\mathbf{x}_j,l_j)\}_{j=1}^{B}$.
3) Model aggregation: server C collects the local gradients from the selected users and then aggregates them to obtain an updated global model. The aggregation is often done by weighted averaging and typically uniform weights are applied, i.e.,

$$\mathbf{G}^{(t)} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{G}_i^{(t)} \qquad (1)$$
$$\mathbf{W}^{(t+1)} = \mathbf{W}^{(t)} - \eta\mathbf{G}^{(t)}.$$

The last two steps are repeated for many iterations until the global model converges or a certain stopping criterion is met.

3.3. Fundamentals of Hidden Subset Sum Problem

Definition 1. ***Hidden Subset Sum Problem (HSSP)*** *Let Q be an integer, and let $x_1, \ldots, x_B$ be integers in $\mathbb{Z}_Q$. Let $\mathbf{a}_1, \ldots, \mathbf{a}_B \in \mathbb{Z}^M$ be vectors with components in $\{0, 1\}$. Let $\mathbf{h} = (h_1, \ldots, h_M) \in \mathbb{Z}^M$ satisfy:*

$$\mathbf{h} = \mathbf{a}_1 x_1 + \mathbf{a}_2 x_2 + \cdots + \mathbf{a}_B x_B \mod Q \qquad (2)$$

Given the modulus Q and the sample vector $\mathbf{h}$, recover the vector $\mathbf{x} = (x_1, \ldots, x_B)$ and the vectors $\mathbf{a}_i$'s, up to a permutation of the x_i's and $\mathbf{a}_i$'s.

We refer to $\mathbf{x}$ as the hidden private data, $\mathbf{a}_i$s as hidden weight vectors, respectively.

Definition 2. ***Hidden Linear Combination Problem (HLCP)*** *Let Q be a positive integer, and let $x_1, \ldots, x_B$ be integers in $\mathbb{Z}_Q$. Let $\mathbf{a}_1, \ldots, \mathbf{a}_B \in \mathbb{Z}^M$ be vectors with components in $\{0, ..., c\}$ for a given $c \in \mathbb{N}^+$. Let $\mathbf{h} = (h_1, \ldots, h_M) \in \mathbb{Z}^M$ satisfy:*

$$\mathbf{h} = \mathbf{a}_1 x_1 + \mathbf{a}_2 x_2 + \cdots + \mathbf{a}_B x_B \mod Q$$

Given Q, c and $\mathbf{h}$, the hidden linear combination problem consists in recovering the vector $\mathbf{x} = (x_1, \ldots, x_B)$ and the vectors $\mathbf{a}_i$'s, up to a permutation of the x_i's and $\mathbf{a}_i$'s.

Note that HLCP is a natural extension of HSSP where the coefficients of the hidden weight vectors $\mathbf{a}_i$' lie in a discrete interval $\{0, ..., c\}$ instead of $\{0, 1\}$, for a given $c \in \mathbb{N}^+$.

3.4. Threat model

We assume that the central server is semi-honest (or passive), it would not change the model parameters maliciously but it can collect information through the learning process. With the collected information, the goal of the central server is to reconstruct the input training samples.

4. Problem formulation

By inspecting (1) we can see that at each iteration t the server C can collect local gradients $\{G_i^{(t)}\}_{i=1,\ldots,N}$. For an individual user $i \in \{1, \ldots, N\}$, denotes $\{(\mathbf{x}_j, l_j)\}_{j=1}^B$ as the mini-batch that produces gradient $G_i^{(t)}$. How much information about the individual training samples $\{(\mathbf{x}_j, l_j)\}_{j=1}^B$ does the gradient $G_i^{(t)}$ carry on depends a lot on the model complexity. Here we assume a multilayer perceptron (MLP) which consists of a number of fully connected layers. Mathematically, a fully connected layer can be described using a linear transformation followed by an activation function (we consider ReLU here). Let u denote the dimension of input $\mathbf{x}_j$ and M denote the number of neurons in the first fully connected layer. Define the weight matrix as $\mathbf{W} \in \mathbb{R}^{M \times u}$ and bias vector as $\mathbf{b} \in \mathbb{R}^M$. For the m^{th} neuron, denote $\mathbf{w}_m^T$ and b_m as the corresponding row in the weight matrix $\mathbf{W}$ and the component in the bias vector $\mathbf{b}$, respectively. Given an input $\mathbf{x}_j$, denote the output of the m^{th} neuron as $y_{m,j}$ given by

$$\begin{aligned} y_{m,j} &= \text{ReLU}(\mathbf{w}_m^T \mathbf{x}_j + b_m) \\ &= \max(0, \mathbf{w}_m^T \mathbf{x}_j + b_m). \end{aligned} \qquad (3)$$

Let $\mathbf{g}_{\mathbf{w}_m^T}$ and g_{b_m} denote the gradient of the loss $\mathcal{L}$ w.r.t the weight $\mathbf{w}_m^T$ and bias b_m, respectively. Given a mini-batch of samples $\{(\mathbf{x}_j, l_j)\}_{j=1}^B$, the gradient $\mathbf{g}_{\mathbf{w}_m^T}$ equals the average of all gradients computed for each of the data points that make up the mini-batch, i.e.,

$$\mathbf{g}_{\mathbf{w}_m^T} = \frac{1}{B} \sum_{j=1}^{B} \frac{\partial \mathcal{L}}{\partial y_{m,j}} \frac{\partial y_{m,j}}{\partial \mathbf{w}_m^T}. \qquad (4)$$

Similarly, g_{b_m} is given by

$$g_{b_m} = \frac{1}{B} \sum_{j=1}^{B} \frac{\partial \mathcal{L}}{\partial y_{m,j}} \frac{\partial y_{m,j}}{\partial b_m}. \qquad (5)$$

4.1. A special case where $B = 1$

It is proven in [17] that a single input data point $\mathbf{x}$ can be reconstructed from the gradients when considering a biased fully-connected layer preceded solely by (possibly unbiased) fully-connected layers. The main idea is very simple. When $B = 1$ and find a neuron where $y_m = \mathbf{w}_m^T \mathbf{x} + b_m$, the above (5) becomes

$$\frac{\partial \mathcal{L}}{\partial b_m} = \frac{\partial \mathcal{L}}{\partial y_m} \frac{\partial y_m}{\partial b_m} = \frac{\partial \mathcal{L}}{\partial y_m}$$

as $\frac{\partial y_m}{\partial b_m} = 1$. Given the fact that $\frac{\partial y_m}{\partial \mathbf{w}_m^T} = \mathbf{x}^T$, (4) becomes

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}_m^T} = \frac{\partial \mathcal{L}}{\partial y_m} \frac{\partial y_m}{\partial \mathbf{w}_m^T} = \frac{\partial \mathcal{L}}{\partial y_m} x^T = \frac{\partial \mathcal{L}}{\partial b_m} \mathbf{x}^T.$$

Hence, if there exists one neuron m such that $\frac{\partial \mathcal{L}}{\partial b_m} \neq 0$, the private data sample x can be perfectly reconstructed using gradients $\frac{\partial \mathcal{L}}{\partial b_m}, \frac{\partial \mathcal{L}}{\partial \mathbf{w}_m^T}$.

As in practice the batch size is rarely 1, in the following we will analyze a more general case for $B > 1$.

4.2. A compact formulation with $B > 1$

For the general case where $B > 1$, to obtain a compact problem formulation, collect the above gradients of all neurons and stack them together we obtain

$$\mathbf{G}_w = \begin{bmatrix} \mathbf{g}_{\mathbf{w}_1^T} \\ \mathbf{g}_{\mathbf{w}_2^T} \\ \vdots \\ \mathbf{g}_{\mathbf{w}_M^T} \end{bmatrix} \in \mathbb{R}^{M \times u}. \qquad (6)$$

$$\mathbf{g}_b = \begin{bmatrix} g_{b_1} \\ g_{b_2} \\ \vdots \\ g_{b_M} \end{bmatrix} \in \mathbb{R}^M. \tag{7}$$

Further stack all data of the mini-batch together we have

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \vdots \\ \mathbf{x}_B^T \end{bmatrix} \in \mathbb{R}^{B \times u}.$$

Define the following matrices:

$$\mathbf{L} = \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial y_{1,1}} & \frac{\partial \mathcal{L}}{\partial y_{1,2}} & \cdots & \frac{\partial \mathcal{L}}{\partial y_{1,B}} \\ \frac{\partial \mathcal{L}}{\partial y_{2,1}} & \frac{\partial \mathcal{L}}{\partial y_{2,2}} & \cdots & \frac{\partial \mathcal{L}}{\partial y_{2,B}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \mathcal{L}}{\partial y_{M,1}} & \frac{\partial \mathcal{L}}{\partial y_{M,2}} & \cdots & \frac{\partial \mathcal{L}}{\partial y_{M,B}} \end{bmatrix} \in \mathbb{R}^{M \times B}.$$

For $k = \{1, 2, \ldots, u\}$ we denote

$$\mathbf{Y}_k = \begin{bmatrix} \frac{\partial y_{1,1}}{\partial w_{1,k}} & \frac{\partial y_{1,2}}{\partial w_{1,k}} & \cdots & \frac{\partial y_{1,B}}{\partial w_{1,k}} \\ \frac{\partial y_{2,1}}{\partial w_{2,k}} & \frac{\partial y_{2,2}}{\partial w_{2,k}} & \cdots & \frac{\partial y_{2,B}}{\partial w_{2,k}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial y_{M,1}}{\partial w_{M,k}} & \frac{\partial y_{M,2}}{\partial w_{M,k}} & \cdots & \frac{\partial y_{M,B}}{\partial w_{M,k}} \end{bmatrix} \in \mathbb{R}^{M \times B}.$$

With (3) we have

$$\frac{\partial y_{m,j}}{\partial w_{m,k}} = \begin{cases} \mathbf{x}_j[k], & \text{if } \mathbf{w}_m^T \mathbf{x}_j + b_m > 0 \\ \mathbf{0}, & \text{otherwise} \end{cases}.$$

Define a binary matrix $\mathbf{R} \in \mathbb{Z}^{M \times B}$ where its (m, j)-th entry is given by

$$r_{m,j} = \begin{cases} 1, & \text{if } \mathbf{w}_m^T \mathbf{x}_j + b_m > 0 \\ 0, & \text{otherwise} \end{cases}.$$

Take the k-th column of $\mathbf{X}$, i.e., $\mathbf{X}[:;k]$, and use it to further define matrix $\mathbf{X}_k$ as

$$\mathbf{X}_k = \begin{bmatrix} \mathbf{X}[:;k]^T \\ \mathbf{X}[:;k]^T \\ \vdots \\ \mathbf{X}[:;k]^T \end{bmatrix} \in \mathbb{R}^{M \times B},$$

thus $\mathbf{Y}_k$ can be represented as

$$\mathbf{Y}_k = \mathbf{X}_k \odot \mathbf{R}$$

Lemma 1. *Let* $\mathbf{A}, \mathbf{R} \in \mathbb{R}^{M \times B}$, $\mathbf{B} \in \mathbb{R}^{B \times M}$. *Then*

$$\mathrm{diag}((\mathbf{A} \odot \mathbf{R}) \cdot \mathbf{B}) = \mathrm{diag}(\mathbf{A} \cdot (\mathbf{B} \odot \mathbf{R}^\top)).$$

Proof. The i-th diagonal element of $(\mathbf{A} \odot \mathbf{R}) \cdot \mathbf{B}$ is

$$\langle \mathbf{A}[i;:] \odot \mathbf{R}[i;:], \mathbf{B}[:;i] \rangle = \langle \mathbf{A}[i;:], \mathbf{B}[:;i] \odot \mathbf{R}^\top[:;i] \rangle,$$

where the right side of the equation is the i-th diagonal element of $\mathbf{A} \cdot (\mathbf{B} \odot \mathbf{R}^\top)$. □

As for (6), we can represent the k-th column of $\mathbf{G}_w$ as

$$\begin{aligned} \mathbf{G}_w[:;k] &= \frac{1}{B} \mathrm{diag}\left(\mathbf{L}\mathbf{Y}_k^T\right) \\ &= \frac{1}{B} \mathrm{diag}\left(\mathbf{L}(\mathbf{X}_k^T \odot \mathbf{R}^T)\right) \\ &\overset{(a)}{=} \frac{1}{B} \mathrm{diag}\left((\mathbf{L} \odot \mathbf{R})\mathbf{X}_k^T\right) \\ &\overset{(b)}{=} \frac{1}{B} (\mathbf{L} \odot \mathbf{R})\mathbf{X}[:;k], \end{aligned} \tag{8}$$

where (a) uses Lemma 1, (b) uses the fact that all columns of $\mathbf{X}_k^T$ are identical.

Denote matrix $\mathbf{D}$ as

$$\mathbf{D} = \mathbf{L} \odot \mathbf{R} \in \mathbb{R}^{M \times B}.$$

Stacking all $\mathbf{G}_w[:;k]$s for all $k \in \{1, 2, \ldots, u\}$ together we thus have

$$\mathbf{G}_w = \frac{1}{B} \mathbf{D}\mathbf{X}. \tag{9}$$

Accordingly, (7) can be represented as

$$\mathbf{g}_b = \frac{1}{B} (\mathbf{L} \odot \mathbf{R}) \mathbf{1}_B = \frac{1}{B} \mathbf{D} \mathbf{1}_B. \tag{10}$$

4.3. Connection to HSSP

Before explaining how the above gradients formulation is connected to HSSP, we first define the following extension of HSSP by considering multi-dimension, since the hidden private data is often multidimensional in the context of machine learning.

Definition 3. ***Multidimensional Hidden Subset Sum Problem (mHSSP)*** *Let Q be an integer, and let $\{\mathbf{x}_i\}_{i \in \{1,\ldots,B\}}$ be vectors in $\mathbb{Z}_Q^u$. Let $\mathbf{a}_1, \ldots, \mathbf{a}_B \in \mathbb{Z}^M$ be vectors with components in $\{0, 1\}$. $\forall j = \{1, \ldots, u\}$, let $\mathbf{h}_j \in \mathbb{Z}^M$ satisfying :*

$$\mathbf{h}_j = \mathbf{a}_1 x_{1,j} + \mathbf{a}_2 x_{2,j} + \cdots + \mathbf{a}_B x_{B,j} \mod Q$$

Given the modulus Q and the sample matrix $\mathbf{H} = [\mathbf{h}_1; \ldots; \mathbf{h}_u] \in \mathbb{Z}^{M \times u}$, recover the vectors $\{\mathbf{x}_i\}_{i \in \{1,\ldots,B\}}$ and the weights vectors $\mathbf{a}_i$'s, up to a permutation of the $\mathbf{x}_i$'s and $\mathbf{a}_i$'s.

Further denote $\mathbf{H} = [\mathbf{h}_1; \ldots; \mathbf{h}_u] \in \mathbb{Z}^{M \times u}$ as the sample matrix, $\mathbf{X} = [\mathbf{x}_1^T, \ldots, \mathbf{x}_B^T] \in \mathbb{Z}^{B \times u}$ as the matrix of hidden private data and $\mathbf{A} = [\mathbf{a}_1; \ldots; \mathbf{a}_B] \in \{0, 1\}^{M \times B}$ as the hidden weight matrix. Thus, the above can be compactly expressed as

$$\mathbf{H} = \mathbf{A}\mathbf{X} \mod Q \tag{11}$$

Accordingly, the definition of multidimensional HLCP (mHLCP) is the same as the above except each entry of the hidden weight matrix $\mathbf{A}$ lies in a discrete interval $\{0, ..., c\}$ instead of $\{0, 1\}$.

Hence, with the multidimensional definition we note that the above-mentioned gradient (9) can be seen as an instance of mHLCP as it has a similar form as (11), where the floating points can be scaled up to integers and the modulo operation will not make a difference if a sufficiently large

Q is considered. To simplify the problem, here we consider the weight in $\mathbf{D}$ is binary, i.e., $\mathbf{L}$ is an all one matrix and $\mathbf{D} = \mathbf{R}$. We will show later even with this simplified setting the mHSSP is still hard to solve under a large bath size B. Hence, in what follows we will analyze the hardness of the input reconstruction attack in FL via analyzing mHSSP, as in the latter there are lots of existing cryptographic results and tools to be exploited.

5. HSSP and attacks

HSSP is an extension of the Subset Sum Problem (SSP) and the hardness of HSSP is not completely understood yet. Note that SSP is a well-known NP-complete problem, thus currently it is unknown if a deterministic polynomial algorithm exists [23]. It is shown that, under certain parameter settings, an HSSP-solver could be used to attack SSP [22], e.g. when an HSSP instance having a unique solution can be generated, solving such HSSP is sufficient for solving SSP. Indeed, in the former, both the mixturing weights and private data are hidden, while in the latter, only the weights are hidden.

In what follows, we first introduce the necessary fundamentals of lattices and then introduce existing lattice attacks for solving HSSP. We first start with the Nguyen-Stern attack (NS attack), the very first HSSP attack raised by Nguyen and Stern [39]. After that we will introduce two advanced attacks based on the NS attack by Coron and Gini based on different principles [20], [21].

5.1. Fundamentals of lattices

A lattice is a discrete subgroup of $\mathbb{R}^m$, which can be defined as follows [40].

Definition 4. ***Lattice*** *Given n linearly independent vectors $\mathbf{b}_1, \ldots, \mathbf{b}_n$ in $\mathbb{R}^m$, the lattice generated by the basis $\mathbf{b}_1, \ldots, \mathbf{b}_n$ is defined as the set*

$$\mathcal{L}(\mathbf{b}_1, \ldots, \mathbf{b}_n) = \left\{ \sum_{i=1}^{d} x_i \mathbf{b}_i | x_i \in \mathbb{Z}, i = 1, \ldots, n \right\},$$

where m denotes the dimension of the lattice $\mathcal{L}$, denoted as $\dim(\mathcal{L})$ and n denotes its rank. If $n = m$, the lattice is called full-rank. $\dim(\mathcal{L})$ means the rank of the lattice $\mathcal{L}$. Define $\mathbf{B}$ as the $m \times n$ base matrix with $\mathbf{b}_1, \ldots, \mathbf{b}_n$ as its columns, then

$$\mathcal{L}(\mathbf{B}) = \mathcal{L}(\mathbf{b}_1, \ldots, \mathbf{b}_n) = \{\mathbf{B} \cdot \mathbf{x} | \mathbf{x} \in \mathbb{Z}^n\}.$$

It could be proved that $\mathcal{L}(\mathbf{B}) = \mathcal{L}(\mathbf{B}')$ if and only if there exists an unimodular matrix $\mathbf{U} \in \mathrm{GL}_d(\mathbb{Z})$ such that $\mathbf{BU} = \mathbf{B}'$. The determinant of the lattice $\mathcal{L}(\mathbf{B})$ is given as $\sqrt{\det(\mathbf{B}^\top \mathbf{B})}$. It could be checked that if $\mathcal{L}(\mathbf{B}) = \mathcal{L}(\mathbf{B}')$, $\det(\mathcal{L}(\mathbf{B})) = \det(\mathcal{L}(\mathbf{B}'))$. From a geometrical point of view, the determinant of a lattice is inversely proportional to its density: the smaller the determinant, the denser the lattice is.

In this paper, we generally consider integer lattices, i.e., the lattices belonging to $\mathbb{Z}^m$.

Definition 5. ***Orthogonal lattice*** *Let $\mathcal{L} \subseteq \mathbb{Z}^m$ be a lattice. Its orthogonal lattice is given as*

$$\mathcal{L}^\perp := \{\mathbf{y} \in \mathbb{Z}^m | \forall \mathbf{x} \in \mathcal{L}, \langle \mathbf{x}, \mathbf{y} \rangle = 0\},$$

where $\langle \ , \ \rangle$ is inner product of $\mathbb{R}^m$. Its orthogonal lattice modulo Q is given as

$$\mathcal{L}_Q^\perp := \{\mathbf{y} \in \mathbb{Z}^m | \forall \mathbf{x} \in \mathcal{L}, \langle \mathbf{x}, \mathbf{y} \rangle \equiv 0 \mod Q\}.$$

The completion of a lattice $\mathcal{L}$ is the lattice $\bar{\mathcal{L}} = Span_{\mathbb{R}}(\mathcal{L}) \cap \mathbb{Z}^m = (\mathcal{L}^\perp)^\perp$, where $Span_{\mathbb{R}}(\mathcal{L}) = \{\mathbf{B} \cdot \mathbf{x} | \mathbf{x} \in \mathbb{R}^n\}$ for $\mathcal{L} = \mathcal{L}(\mathbf{B})$. Note that $(\mathcal{L}^\perp)^\perp$ contains the original lattice $\mathcal{L}$.

Theorem 1. *Let $\mathcal{L} \subseteq \mathbb{Z}^m$ be a lattice. Then $\dim(\mathcal{L}^\perp) + \dim(\mathcal{L}) = m$.*

Definition 6. ***Shortest vector*** *Let $\|\cdot\|$ denote the Euclidean norm. Then we could find a non-zero vector $\mathbf{v}$ of the minimal norm in every lattice $\mathcal{L}$. The first minimum of $\mathcal{L}$ is $\lambda_1(\mathcal{L}) = \|\mathbf{v}\|$, and $\mathbf{v}$ is called the shortest vector.*

The following is a generalization of λ_1.

Definition 7. ***Successive minima*** *Let $\mathcal{L}$ be a lattice of rank n. For $i \in \{1, \ldots, n\}$, the i-th successive minimum $\lambda_i(\mathcal{L})$ is defined by the minimum of maximum norm of any i linearly independent lattice points.*

The so-called LLL and BKZ algorithms are important building blocks for designing HSSP attacks. A brief introduction about them is given as follows:

- **LLL**: The LLL algorithm [41] is a polynomial time lattice reduction algorithm with a basis of a lattice $\mathcal{L}$ as input and output a basis which is LLL-reduced, meaning that the basis is short and nearly orthogonal.
- **BKZ**: The BKZ algorithm [42] is also a lattice reduction algorithm. Increasing the block-size parameter of the algorithm improves the accuracy, at the cost of a longer computation time. Namely, BKZ-2 can produce an LLL-reduced basis in polynomial time, while with full block-size one can retrieve the shortest vector of the lattice in exponential time.

5.2. NS attack

The whole process of the NS attack could be divided into two steps:

- **Step 1**: The first step is based on orthogonal lattice attack. As the only known information in (2) is $\mathbf{h}$ (in Section 5.4 we will discuss how to extend to the multidimensional case for (11)), the orthogonal lattice attack starts with computing the orthogonal lattice of $\mathbf{h}$ modulus Q, i.e., $\mathcal{L}_Q^\perp(\mathbf{h})$. As $\mathbf{h}$ is a linear combination of the hidden weight vectors $\{\mathbf{a}_i\}_{i \in \{1,\ldots,B\}}$, the orthogonal lattice of $\mathbf{A}$, i.e., $\mathcal{L}^\perp(\mathbf{A})$, is then contained in $\mathcal{L}_Q^\perp(\mathbf{h})$. Based on the fact that $\mathbf{a}_i$'s are binary, the goal is to ensure the first $M - B$ short vectors of the LLL basis of $\mathcal{L}_Q^\perp(\mathbf{h})$

form a basis of $\mathcal{L}^{\perp}(\mathbf{A})$. After obtaining $\mathcal{L}^{\perp}(\mathbf{A})$, then compute its orthogonal $(\mathcal{L}^{\perp}(\mathbf{A}))^{\perp}$ using the LLL algorithm again, as $(\mathcal{L}^{\perp}(\mathbf{A}))^{\perp}$ contains our target lattice $\mathcal{L}(\mathbf{A})$. Note that the LLL algorithm is applied twice. The first time is to compute the LLL-reduced basis of $\mathcal{L}_Q^{\perp}(\mathbf{h})$; the second time is to compute the orthogonal lattice of $\mathcal{L}^{\perp}(\mathbf{A})$ [43], [44].

- **Step 2**: The second step is to recover the binary vectors $\mathbf{a}_i$' from a LLL-reduced basis of $(\mathcal{L}^{\perp}(\mathbf{A}))^{\perp}$, after that the hidden private data vector $\mathbf{x}$ can then be recovered. Since the short vectors found by Step 1 may not be short enough, thus the BKZ algorithm is applied in Step 2 to find shorter vectors. Denote $\{\mathbf{v}_i\}_{i=1}^{M-B}$ as the obtained short vectors in $(\mathcal{L}^{\perp}(\mathbf{A}))^{\perp}$. Given the fact that $\mathbf{a}_i$'s are binary, it could be proved with high possibility that $\mathbf{v}_i$s are either $\{\mathbf{a}_i\}$'s or $\{\mathbf{a}_i - \mathbf{a}_j\}$'s. Assume that the $\mathbf{a}_i$'s are the only binary vectors in $(\mathcal{L}^{\perp}(\mathbf{A}))^{\perp}$, then take B binary vectors in $\{\mathbf{v}_i\} \cup \{\mathbf{v}_i - \mathbf{v}_j\} \cup \{\mathbf{v}_i + \mathbf{v}_j\}$ as $\mathbf{a}_i$'s. After obtaining $\mathbf{A}$, pick up a $B \times B$ sub-matrix $\mathbf{A}'$ from $\mathbf{A}$ with non-zero determinant modulus Q. Then $\mathbf{A}' \cdot \mathbf{x} \equiv \mathbf{h}' \mod Q$. Hence, the hidden private data can be recovered as $\mathbf{x} \equiv \mathbf{A}'^{-1}\mathbf{h}' \mod Q$.

The strategy described above is guaranteed to work with a good probability if the parameters M, B, Q satisfy certain conditions. The core intuition behind those conditions is that the short vectors of the LLL basis of $\mathcal{L}_Q^{\perp}(\mathbf{h})$ should form a basis of $\mathcal{L}^{\perp}(\mathbf{A})$. Let the vector $\mathbf{y}$ be orthogonal modulo Q to $\mathbf{h}$. We thus have

$$\langle \mathbf{y}, \mathbf{h} \rangle = x_1 \langle \mathbf{y}, \mathbf{a}_1 \rangle + \cdots + x_B \langle \mathbf{y}, \mathbf{a}_B \rangle \equiv 0 \mod Q,$$

which implies that the vector

$$\mathbf{p_y} = (\langle \mathbf{y}, \mathbf{a}_1 \rangle, \ldots, \langle \mathbf{y}, \mathbf{a}_B \rangle)$$

is orthogonal to the hidden private data vector $\mathbf{x} = (x_1, \ldots, x_B)$, i.e., $\mathbf{p_y} \in \mathcal{L}_Q^{\perp}(\mathbf{x})$. It follows that if the norm of $\mathbf{p_y}$ is less than the first minimum of $\mathcal{L}_Q^{\perp}(\mathbf{x})$, $\mathbf{p_y}$ must be a zero vector. Hence, $\mathbf{y} \in \mathcal{L}^{\perp}(\mathbf{A})$. Since $\dim(\mathcal{L}^{\perp}(\mathbf{A})) = M - B$, let an upper bound of the $M - B$ successive minimum of $\mathcal{L}^{\perp}(\mathbf{A})$ is less than an estimation of the lower bound of the first minimum of $\mathcal{L}_Q^{\perp}(\mathbf{x})$. To guarantee $\mathbf{y} \in \mathcal{L}^{\perp}(\mathbf{A})$, the following inequality is required:

$$\log Q > \iota MB + \frac{MB}{2(M-B)} \log M + \frac{B}{2} \log(M - B),$$

where $0 < \iota < 1$ is decided by the so-called LLL Hermite factor which controls the quality of the LLL-reduced basis. For more details, we refer the readers to [22].

5.3. CG attacks

Recall that the BKZ algorithm is applied to find shorter vectors in order to recover the binary vectors in Step 2, thus the complexity of the NS attack algorithm is not polynomial time. Coron and Gini [20], [21] have put forward two alternatives of Step 2, which are able to recover the binary vectors $\mathbf{a}_i$'s within a polynomial time.

The multivariate approach [20]: Instead of using the BKZ algorithm to recover the shorter vectors, the multivariate approach proposes to recover the hidden vectors $\mathbf{a}_i$'s using the multivariate quadratic polynomial system. Let $\{\mathbf{u}_j\}_{j \in \{1,\ldots,B\}}$ denote a basis of $(\mathcal{L}^{\perp}(\mathbf{A}))^{\perp}$ as the output of the NS algorithm's Step 1. Stacking them together and denote as matrix $\mathbf{U} = [\mathbf{u}_1; \ldots; \mathbf{u}_B] \in \mathbb{Z}^{M \times B}$, further define a matrix $\mathbf{W} \in \mathbb{Z}^{B \times B}$ such that the following holds:

$$\mathbf{UW} = \mathbf{A}. \quad (12)$$

The goal is then to recover the unknown $\mathbf{W}$ and $\mathbf{A}$ based on the knowledge of $\mathbf{U}$. Based on the knowledge that $\mathbf{A}$ is binary, we thus have

$$\mathbf{U}[i; :]\mathbf{W}[:; j] = A[i, j] \in \{0, 1\}.$$

As $0, 1$ are solutions of the quadratic equation $x^2 - x = 0$, the above can thus be written as

$$\langle \mathbf{U}[i; :], \mathbf{W}[:; j] \rangle^2 - \langle \mathbf{U}[i; :], \mathbf{W}[:; j] \rangle = 0,$$

where $i = \{1, \ldots, M\}, j = \{1, \ldots, B\}$. Hence, the above MB quadratic equations form a multivariate quadratic system for unknown $\mathbf{W} \in \mathbb{Z}^{B \times B}$. Solving the quadratic system we can obtain $\mathbf{W}$, thereby recovering $\mathbf{A}$.

We note that the above multivariate quadratic polynomial system requires that $M \approx B^2$ instead of $M = 2B$ in the NS algorithm. With a bigger M, the complexity of Step 1 would be larger (see time complexity analysis in Section 6.1). To avoid the high complexity, the authors proposed a blocks orthogonal lattice attack to reduce the complexity of Step 1 to $\mathcal{O}(B^9)$.

The statistical approach [21]: Recall (12), since the ranks of $\mathbf{U}$ and $\mathbf{A}$ are both B, thus $\mathbf{W}$ is invertible over $\mathbb{Q}$. Denote its inverse as matrix $\mathbf{V} = \mathbf{W}^{-1}$. We then have

$$\mathbf{U} = \mathbf{A} \cdot \mathbf{V}.$$

We remark that the above has a similar form of the so-called continuous parallelepiped, which is defined as

$$\mathcal{P}_{[-1,1]}(\mathbf{V}) = \{\sum_{i=1}^{B} c_i \mathbf{V}[i; :] | c_i \in [-1, 1]\},$$

where $\mathbf{V} \in \mathrm{GL}_B(\mathbb{R})$. Given the fact that $\mathbf{A}$ is binary, thus $\mathbf{A} \cdot \mathbf{V}[i; :]$ belongs to the following discrete parallelepiped with binary weights:

$$\mathcal{P}_{\{0,1\}}(\mathbf{V}) = \{\sum_{i=1}^{B} c_i \mathbf{V}[i; :] | c_i \in \{0, 1\}\}.$$

Hence,

$$\{\mathbf{U}[i; :]\}_{i=\{1,\ldots,M\}} \subset \mathcal{P}_{\{0,1\}}(\mathbf{V}) \subset \mathcal{P}_{[-1,1]}(\mathbf{V}).$$

There is an existing algorithm called Nguyen-Regev algorithm [45] which is able to return with probability a good approximation of a row of $\mathbf{V}$, given $poly(n)$ vectors uniformly sampled form $\mathcal{P}_{[-1,1]}(\mathbf{V})$. Coron and Gini first apply the Nguyen-Regev algorithm and then conduct some modifications such that the target binary $\mathbf{A}$ can be recovered.

M	$\log Q$	First LLL	Second LLL
$\gg B$	$\mathcal{O}(BM)$	$\mathcal{O}(M^7 \cdot B^2)$	$\mathcal{O}(M^9/B^2)$
B^2	$\mathcal{O}(B^3)$	$\mathcal{O}(B^{16})$	$\mathcal{O}(B^{16})$
$2B$	$\mathcal{O}(B^2)$	$\mathcal{O}(B^9)$	$\mathcal{O}(B^7)$
$B+1$	$\mathcal{O}(B^2 \log B)$	$\mathcal{O}(B^9 \log^2 B)$	$\mathcal{O}(B^7 \log^2 B)$

TABLE 1: Modulus size and time complexity of the two LLL algorithms of Step 1.

5.4. Optimized attack for mHSSP

As the input of FL is often multidimensional, thus we modify the above attack to exploit this property. One modified way is to change $\mathcal{L}_Q^{\perp}(\mathbf{h})$ to $\mathcal{L}_Q^{\perp}(\mathbf{H})$. In existing attacks, the orthogonal lattice $\mathcal{L}_Q^{\perp}(\mathbf{h})$ is constructed as follows:

Recall h_1 denotes the first entry of vector $\mathbf{h}$, for simplicity we assume the greatest common divisor $\gcd(h_1, Q) = 1$ (see Appendix A for a more generalized discussion). In addition, $\langle (Q, 0, \ldots, 0), \mathbf{h} \rangle \equiv 0 \mod Q$. Define $\mathbf{h}' = (h_2, ..., h_M)$ and a basis of $\mathcal{L}_Q^{\perp}(\mathbf{h})$ can be constructed as

$$\begin{bmatrix} Q & \mathbf{0} \\ -\mathbf{h}' \cdot (h_1^{-1} \mod Q) & \mathbf{I}_{M-1} \end{bmatrix}^T,$$

as it can be verified that each column vector is orthogonal to $\mathbf{h}$ modulo Q.

Note that in practice, the dimension of the hidden private data u can be very large, for example $u = 3072$ using CIFAR-100 [46] as the training dataset, thus sometimes it is not necessary to use all u columns for reconstruction. we can take a few columns, say $r \leq u$, to construct $\mathcal{L}_Q^{\perp}(\mathbf{H})$, i.e. the column dimension of $\mathbf{H}$ is r. For simplicity, assume $\mathbf{H}[1, \ldots, r; :] \in \mathbb{Z}^{r \times r}$ is invertible modulus Q and denote the inverse as $\mathbf{H}_r^{-1}$. Hence, we then construct a basis of $\mathcal{L}_Q^{\perp}(\mathbf{H})$ with the following:

$$\begin{bmatrix} Q\mathbf{I}_r & \mathbf{0} \\ -\mathbf{H}[r+1, \ldots, M; :] \cdot \mathbf{H}_r^{-1} & \mathbf{I}_{M-r} \end{bmatrix}^T.$$

After constructing $\mathcal{L}_Q^{\perp}(\mathbf{H})$, the rest procedures of mHSSP attacks are the same as the original HSSP attacks.

6. Attack complexity and defense

In this section, we first introduce the time complexity of the above HSSP attacks, then we briefly explain a popular defense called secure data aggregation and analyze its time complexity accordingly.

6.1. Complexity analysis

In Table 1 the modulus size of the HSSP and time complexity of the first LLL algorithm and second LLL algorithm Step 1 are summarized [22], recall that M is the number of neurons in the first fully connected layer, which reflects the model complexities. For the NS algorithm, the whole complexity may not be polynomial time. The improved multivariate and statistical algorithms are in polynomial time. Note that we only show the time complexity of the two LLL algorithms in Step 1 because of the following reasons: 1) all three attacks share the same Step 1; 2) in the experiments it is shown that the main bottleneck of the running time using the recent multivariate and statistical attacks, lies in Step 1 as the time consumed in Step 2 grows slower than Step 1 when increasing the batch size B. As for M, we can see that it does not dominate the time complexity unless $M \gg B$ (this can be avoided as one can always select to use partial information if M is too big). Overall, we conclude that we can expect the whole time complexity to be $\mathcal{O}(B^9)$ when the input data coefficients are small compared to Q, e.g. $[0, 255]$ for the case of image inputs.

6.2. Applying secure data aggregation before sharing gradients

There are various ways to enhance the privacy of FL, one of the most widely-adopted methods is to apply secure data aggregation techniques to securely compute the average of the local gradients without revealing them [47]. Popular secure data aggregation techniques include secure multiparty computation [24], which allows multiple parties to jointly compute the output of a function using their private inputs without revealing each individual private input, and homomorphic encryption [25], [48], [49], which guarantees privacy by conducting computation only on encrypted domains.

After applying secure data aggregation, the available information to the server is thus the global average of the local gradients of all clients. Assuming there are N clients, let $\mathbf{G}_{w,i}$, $\mathbf{D}_i$ and $\mathbf{X}_i$ denotes the local gradient, the hidden weight matrix, and input training data of client i where $i = \{1, \ldots, N\}$, respectively. The global average of the local gradients of (9), denoted as $\mathbf{G}_{w,\text{ave}}$, can then be represented as,

$$\begin{aligned} \mathbf{G}_{w,\text{ave}} &= \frac{1}{N} \sum_{i=1}^{N} \mathbf{G}_{w,i} \\ &= \frac{1}{NB} \begin{bmatrix} \mathbf{D}_1; & \mathbf{D}_2; & \ldots; & \mathbf{D}_N \end{bmatrix} \begin{bmatrix} \mathbf{X}_1 \\ \mathbf{X}_2 \\ \vdots \\ \mathbf{X}_N \end{bmatrix}. \end{aligned}$$

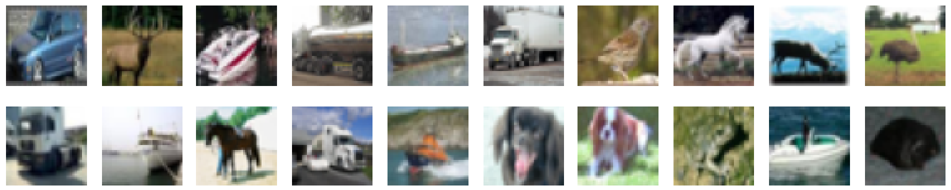
That is, applying secure data aggregation implies that the hidden inputs row dimension increases by N times. Hence, the attack complexity directly increases from $\mathcal{O}(B^9)$ to $\mathcal{O}(N^9 B^9)$.

7. Numerical results and discussions

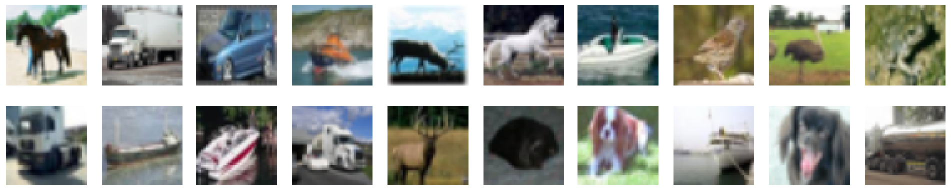
We now proceed to demonstrate the numerical results for mHSSP attacks.

7.1. Numerical results

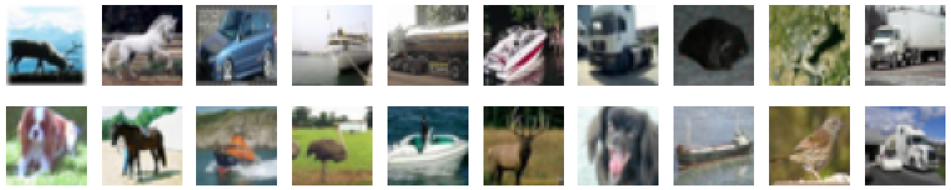
Attack results: To demonstrate the attack results using multidimensional inputs, we choose to use training samples of



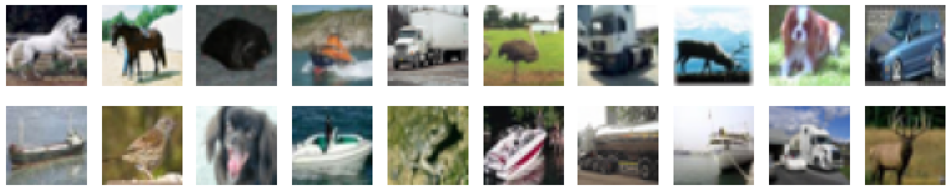
(a) 20 training samples randomly selected from CIFAR-10.



(b) Reconstructed training samples using multidimensional NS attack.



(c) Reconstructed training samples using multidimensional multivariate attack.



(d) Reconstructed training samples using multidimensional statistical attack.

Figure 1: Input reconstruction results of three different mHSSP attacks with $B = 20$ samples randomly selected from CIFAR-10 dataset.

CIFAR-10 dataset where the dimension of $u = 32 \times 32 \times 3 = 3072$. In Fig. 1 we show the reconstructed samples using the three described mHSSP attacks under the case of $B = 20$. More specifically, we set $M = 300$ and the hidden weight matrix $\mathbf{A}$ is randomly generated with uniform probability. As we can see, all mHSSP attacks can successfully recover the original input private samples, except the fact that the order is permuted. Note that in this case, the reconstruction is perfect, i.e., the Euclidean distance of the reconstructed samples and the original samples is zero.

To get a feeling on how fast does the time complexity grow with the batch size, in Fig. 2 we demonstrate the running time in terms of the batch size B using the three attacks. There we can see that, as expected, the recent Multivariate and Statistical attacks are faster than NS attack. While, the running time grows very fast when increasing the batch size.

No restrictive assumption for samples with repeated labels: As mentioned before, existing gradient inversion attacks often require that the samples within a mini-batch should not have repeated labels [9], otherwise the reconstructed images for the same label would be quite similar. We remark that this restrictive assumption is not required in HSSP. To validate this, we randomly select 20 samples from the same 'cat' label of CIFAR-10, and test the input reconstruction results using the three mHSSP attacks. Though all images are from the same class, all the input samples can be reconstructed perfectly except for the fact that their order is permuted (see Fig. 3 in Appendix B). Hence, this confirms the benefits of using HSSP to analyze the privacy leakage in FL.

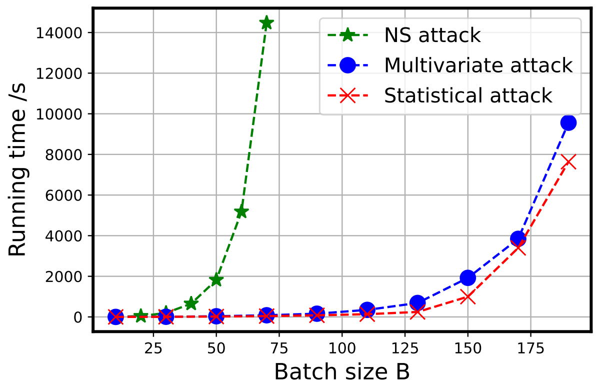


Figure 2: Running time as a function of the batch size B using three existing HSSP attacks including the NS, multivariate and statistical attack.

7.2. Discussions

We remark that our paper is an initial work to analyze the privacy leakage of gradient sharing in FL using HSSP, there are many interesting directions to further explore. A few of them are listed below:

1) Develop more optimized attacks using the bias information provided in (10): Though the bias information is not directly computed using the input training data but it contains information about the weights of each row of the hidden weight matrix $\mathbf{D}$. This information is currently not investigated in existing attacks, it is thus very potential to design a more optimized attack by exploiting it.
2) Exploit the time complexity of attacks using the dimension parameter u: In Section 5.4 we have shown one way to deal with the multidimensional property for constructing $\mathcal{L}_Q^{\perp}(\mathbf{H})$. So far this parameter is not investigated in existing attacks for analyzing the time complexity, which is also very crucial to further analyze as in machine learning the dimension u is often quite large.
3) Analyze the hardness under more complex settings: As mentioned in Section 4.3, we only consider a simplified setting where the weight matrix $\mathbf{D}$ is binary thus it reduces to mHSSP. While it is more practical to consider the case where $\mathbf{D}$ is non-binary, it thus reduces to mHLCP where the weights are within the range of $[0, c]$ (recall Definition 2). We suspect that the time complexity might be much higher than mHSSP as in practice the weight range c can be very large.

8. Conclusion

In this paper, we took the first step to theoretically analyze the problem of input reconstruction attack in FL from a cryptographic view. By mathematically formulating the gradients shared in an MLP network as a classical cryptographic problem called HSSP, we adopted existing HSSP tools to analyze it. Via HSSP attacks, we analytically showed that the time complexity of input reconstruction attack in FL increases as the batch size increases. This analytical result explains why the reconstruction results of existing gradient inversion attacks become very poor when considering a big batch size. In addition, we showed that adopting secure aggregation techniques would be a good defense to input reconstruction attack as it directly increases the time complexity of existing HSSP attacks by N^9 times where N is the number of participants in FL. We also discussed some interesting future directions for further exploration. As an initial work on theoretically analyzing the input reconstruction attack in FL by exploiting cryptographic tools, our work calls for more researchers to explore this exciting cross-disciplinary topic.

References

[1] M. Anderson, *Technology device ownership, 2015*, Pew Research Center, 2015.

[2] J. Poushter and others, “Smartphone ownership and internet usage continues to climb in emerging economies,” *Pew Research Center*, vol. 22, pp. 1–44, 2016.

[3] F. X. Yu P. Richtárik A. T. Suresh J. Konecný, H. B. McMahan and D. Bacon, “Federated learning: Strategies for improving communication efficiency,” *arXiv preprint arXiv:1610.05492*, 2016.

[4] D. Ramage J. Konecnỳ, H. B. McMahan and P. Richtárik, “Federated optimization: Distributed machine learning for on-device intelligence,” *arXiv preprint arXiv:1610.02527*, 2016.

[5] D. Ramage H. B. McMahan, E. Moore and B. A. y Arcas, “Federated learning of deep networks using model averaging,” *arXiv preprint arXiv:1602.05629*, vol. 2, 2016.

[6] L. Zhu, Z. Liu, and S. Han, “Deep leakage from gradients,” *Advances in neural information processing systems*, vol. 32, 2019.

[7] B. Zhao, KR Mopuri, and H. Bilen, “iDLG: Improved deep leakage from gradients,” *arXiv preprint arXiv:2001.02610*, 2020.

[8] J. Geiping. H. Bauermeister, H. Dröge and M. Moeller, “Inverting gradients-how easy is it to break privacy in federated learning?,” *NeurIPS*, vol. 33, pp. 16937–16947, 2020.

[9] H. Yin, A. Mallya, A. Vahdat, JM Alvarez, J. Kautz, and P. Molchanov, “See through gradients: Image batch recovery via gradinversion,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 16337–16346.

[10] F. Boenisch, A. Dziedzic, R. Schuster, AS. Shamsabadi, I. Shumailov, and N. Papernot, “When the curious abandon honesty: Federated learning is not private,” *arXiv preprint arXiv:2112.02918*, 2021.

[11] J. Geng, Y. Mou, Q. Li, F. Li, O. Beyan, S. Decker, and C. Rong, “Improved gradient inversion attacks and defenses in federated learning,” *IEEE Transactions on Big Data*, 2023.

[12] L. Fowl, J. Geiping, W. Czaja, M. Goldblum, and T. Goldstein, “Robbing the fed: Directly obtaining private data in federated learning with modified models,” *arXiv preprint arXiv:2110.13057*, 2021.

[13] J. Zhu and M. Blaschko, “R-gap: Recursive gradient attack on privacy,” *arXiv preprint arXiv:2010.07733*, 2020.

[14] W. Wei, L. Liu, M. Loper, Ka-Ho Chow, ME Gursoy, S. Truex, and Y. Wu, “A framework for evaluating gradient leakage attacks in federated learning,” *arXiv preprint arXiv:2004.10397*, 2020.

[15] H. Yang, M. Ge, K. Xiang, and J. Li, “Using highly compressed gradients in federated learning for data reconstruction attacks,” *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 818–830, 2022.

[16] Z. Zhao, M. Luo, and W. Ding, "Deep leakage from model in federated learning," *arXiv preprint arXiv:2206.04887*, 2022.

[17] J. Xu, C. Hong, J. Huang, L. Y Chen, and J. Decouchant, "Agic: Approximate gradient inversion attack on federated learning," in *2022 41st International Symposium on Reliable Distributed Systems (SRDS)*. IEEE, 2022, pp. 12–22.

[18] A. Wainakh, E. Zimmer, S. Subedi, J. Keim, T. Grube, S. Karuppayah, G. S. Alejandro, and M. Mühlhäuser, "Federated learning attacks revisited: A critical discussion of gaps, assumptions, and evaluation setups," *Sensors*, vol. 23, no. 1, pp. 31, 2022.

[19] AF Agarap, "Deep learning using rectified linear units (relu)," *arXiv preprint arXiv:1803.08375*, 2018.

[20] J. Coron and A. Gini, "A polynomial-time algorithm for solving the hidden subset sum problem," in *Advances in Cryptology–CRYPTO 2020*. Springer, 2020, pp. 3–31.

[21] J. Coron and A. Gini, "Provably solving the hidden subset sum problem via statistical learning," *Mathematical Cryptology*, vol. 1, no. 2, pp. 70–84, 2021.

[22] A. Gini, *On the hardness of the hidden subset sum problem: algebraic and statistical attacks*, Ph.D. thesis, University of Luxembourg, Luxembourg, 2022.

[23] S. Dasgupta, C. H. Papadimitriou, and U. V. Vazirani, *Algorithms*, McGraw-Hill Higher Education New York, 2008.

[24] R. Cramer, I. B. Damgård, and J. B. Nielsen, *Secure Multiparty Computation and Secret Sharing*, Cambridge University Press, 2015.

[25] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *EUROCRYPT, pp. 223–238*, 1999.

[26] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *in Proc. IEEE Symp. Secur. Privacy (SP), pp.3–18*, 2017.

[27] A. Salem, Y. Zhang, M. Humbert, P. Berrang, M. Fritz, and M. Backes, "Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models," *Network and Distributed Systems Security Symposium*, 2019.

[28] S. Yeom, I. Giacomelli, M. Fredrikson, and S. Jha, "Privacy risk in machine learning: Analyzing the connection to overfitting," in *IEEE computer security foundations symposium (CSF)*, 2018, pp. 268–282.

[29] L. Song and P. Mittal, "Systematic evaluation of privacy risks of machine learning models," in *USENIX Security Symposium (USENIX)*, 2021, pp. 2615–2632.

[30] CA Choquette-Choo, F. Tramèr, N. Carlini, and N. Papernot, "Label-only membership inference attacks," in *ICML*. PMLR, 2021, pp. 1964–1974.

[31] G. Ateniese, LV Mancini, A. Spognardi, A. Villani, D. Vitali, and G. Felici, "Hacking smart machines with smarter ones: How to extract meaningful data from machine learning classifiers," *International Journal of Security and Networks*, vol. 10, no. 3, pp. 137–150, 2015.

[32] G. Karan, Q. Wang, W. Yang, CA Gunter, and N. Borisov, "Property inference attacks on fully connected neural networks using permutation invariant representations," in *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*, 2018, pp. 619–633.

[33] M. Fredrikson, E. Lantz, S. Jha, S. Lin, D. Page, and T. Ristenpart, "Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing," in *23rd {USENIX} Security Symposium*, 2014, pp. 17–32.

[34] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, 2015, pp. 1322–1333.

[35] X. Li M. Xu, "Subject property inference attack in collaborative learning," in *2020 12th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC)*. IEEE, 2020, vol. 1, pp. 227–231.

[36] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov, "Exploiting unintended feature leakage in collaborative learning," in *2019 IEEE symposium on security and privacy (SP)*. IEEE, 2019, pp. 691–706.

[37] A. Wainakh, F. Ventola, T. Müßig, J. Keim, C. G. Cordero, E. Zimmer, T. Grube, K. Kersting, and M. Mühlhäuser, "User label leakage from gradients in federated learning," *arXiv preprint arXiv:2105.09369*, 2021.

[38] I. Shumailov, Z. Shumaylov, D. Kazhdan, Y. Zhao, N. Papernot, MA Erdogdu, and RJ Anderson, "Manipulating sgd with data ordering attacks," *Advances in Neural Information Processing Systems*, vol. 34, pp. 18021–18032, 2021.

[39] P. Nguyen and J. Stern, "The hardness of the hidden subset sum problem and its cryptographic implications," in *Crypto*. Springer, 1999, vol. 99, pp. 31–46.

[40] JW Cassels, *An introduction to the geometry of numbers (Reprint)*, Classics in mathematics. Springer-Verlag, Berlin, 1997.

[41] AK Lenstra, HW Lenstra, and L. Lovász, "Factoring polynomials with rational coefficients," *Mathematische annalen*, vol. 261, no. ARTICLE, pp. 515–534, 1982.

[42] Y. Chen and PQ Nguyen, "Bkz 2.0: Better lattice security estimates," in *Advances in Cryptology–ASIACRYPT 2011: 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings 17*. Springer, 2011, pp. 1–20.

[43] P. Nguyen and J. Stern, "Merkle-hellman revisited: a cryptanalysis of the qu-vanstone cryptosystem based on group factorizations," in *Advances in Cryptology—CRYPTO'97: 17th Annual International Cryptology Conference Santa Barbara, California, USA August 17–21, 1997 Proceedings*. Springer, 2006, pp. 198–212.

[44] J. Chen, D. Stehlé, and G. Villard, "Computing an lll-reduced basis of the orthogonal latice," in *Proceedings of the 2018 ACM International Symposium on Symbolic and Algebraic Computation*, 2018, pp. 127–133.

[45] P. Q Nguyen and O. Regev, "Learning a parallelepiped: Cryptanalysis of ggh and ntru signatures," in *Advances in Cryptology-EUROCRYPT 2006: 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28-June 1, 2006. Proceedings 25*. Springer, 2006, pp. 271–288.

[46] K. Alex, H. Geoffrey, et al., "Learning multiple layers of features from tiny images," 2009.

[47] G. Xu, H. Li, S. Liu, K. Yang, and X. Lin, "Verifynet: Secure and verifiable federated learning," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 911–926, 2019.

[48] I. Damgård, V. Pastro, N. Smart, and S. Zakarias, "Multiparty computation from somewhat homomorphic encryption," in *Advances in Cryptology–CRYPTO, pp. 643–662*. Springer, 2012.

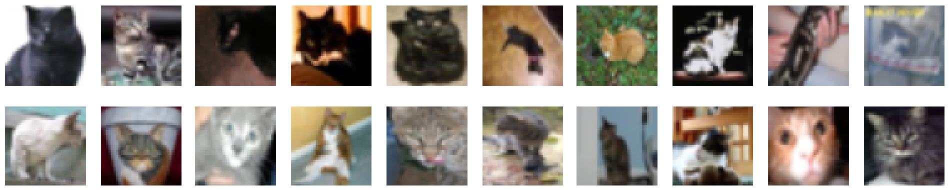
[49] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *STOC, pp.169–178*, 2009.

Appendix A. A general way to construct a basis of $\mathcal{L}_Q^{\perp}(\mathbf{h})$

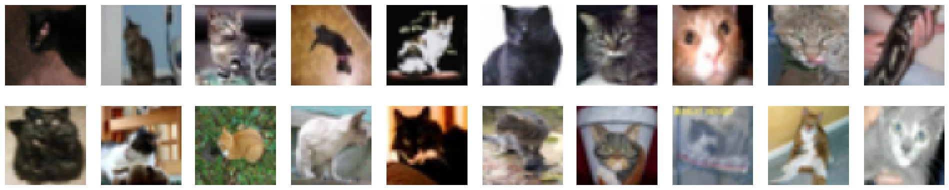
Besides the case of $\gcd(h_1, Q) = 1$ discussed before, in what follows we discuss how to construct the basis of $\mathcal{L}_Q^{\perp}(\mathbf{h})$ for other cases:

The case of $\mathbf{h} \equiv \mathbf{0} \mod Q$: in this case the basis can be directly constructed as $\mathcal{L}_Q^{\perp}(\mathbf{h}) = \mathbb{Z}^M$.

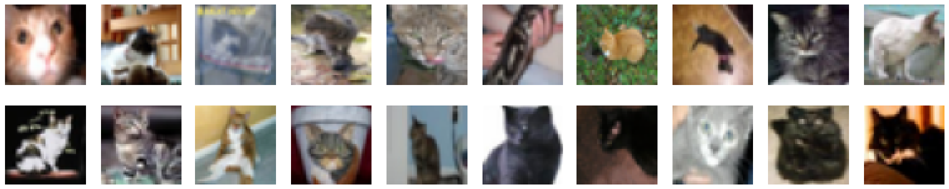
The case of $\gcd(h_1, Q) = \gcd(h_1, h_2, \cdots, h_M, Q) = d$ where $d \neq 1$: the construction is the same as the case of $\gcd(h_1, Q) = 1$ by simply replacing Q and $\mathbf{h}$ with Q/d and $\mathbf{h}/d$, respectively. Hence, the basis of $\mathcal{L}_Q^{\perp}(\mathbf{h})$ can be constructed using the same basis of $\mathcal{L}_{Q/d}^{\perp}(\mathbf{h}/d)$.



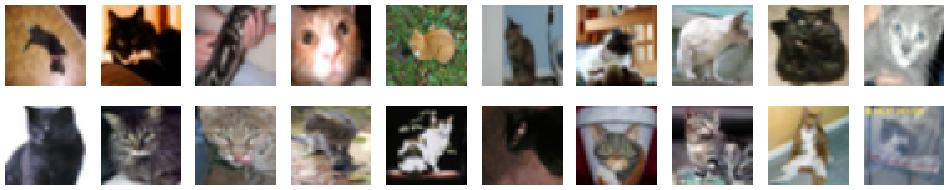
(a) 20 'cat' samples randomly selected from the training set of CIFAR-10.



(b) Reconstructed training samples using the multidimensional NS attack.



(c) Reconstructed training samples using the multidimensional multivariate attack.



(d) Reconstructed training samples using the multidimensional statistical attack.

Figure 3: Input reconstruction results using three different mHSSP attacks for the case that all samples within the mini-batch are from the same class label.

The case of $\gcd(h_1, h_2, \cdots, h_M, Q) = d$ **while** $\gcd(h_i, Q) = d_i \neq d, \ \forall i \in \{1, \ldots, M\}$: define the first vector as $\mathbf{y}_1 := (Q/d_1, 0, \ldots, 0)$, the second one is constructed as $\mathbf{y}_2 := (-(h_2/d_{12}) \cdot ((h_1/d_1)^{-1} \mod Q/d_1), d_1/d_{12}, 0, \ldots, 0)$ where $d_{12} = \gcd(d_1, d_2)$. This construction ensures $\langle \mathbf{y}_2, \mathbf{h} \rangle \equiv 0 \mod Q$ as

$$\begin{aligned}\langle \mathbf{y}_2, \mathbf{h} \rangle &\equiv -(h_2/d_{12}) \cdot ((h_1/d_1)^{-1} \mod Q/d_1) \cdot h_1 \\ &\quad + d_1 h_2/d_{12} \\ &\equiv -(h_2/d_{12}) \cdot ((h_1/d_1)^{-1} \mod Q/d_1) \cdot (h_1/d_1) \cdot d_1 \\ &\quad + d_1 h_2/d_{12} \\ &\equiv -(h_2/d_{12}) \cdot d_1 + d_1 h_2/d_{12} \\ &\equiv 0 \mod Q.\end{aligned}$$

The third vector can be constructed as $\mathbf{y}_3 := (-(h_3/d_{123}) \cdot ((h_1/d_1)^{-1} \mod Q/d_1) \cdot k_1, -(h_3/d_{123}) \cdot ((h_2/d_2)^{-1} \mod Q/d_2) \cdot k_2, d_{12}/d_{123}, 0, \ldots, 0)$ where $d_{123} = \gcd(d_1, d_2, d_3)$ and $k_1, k_2 \in \mathbb{Z}$ satisfying $k_1 d_1 + k_2 d_2 = d_{12}$. It could be checked that $\langle \mathbf{y}_3, \mathbf{h} \rangle \equiv 0 \mod Q$. Hence, the rest can be constructed similarly and collect them together as $[\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_M]$, which is a basis of $\mathcal{L}_Q^{\perp}(\mathbf{h})$.

Appendix B. Attack results for the case of repeated labels

See Fig. 3 for the input reconstruction results of the case where input samples in one mini-batch share the same label.