Protection against Source Inference Attacks in Federated Learning using Unary Encoding and Shuffling

Andreas Athanasiou, Kangsoo Jung, Catuscia Palamidessi

Poster: Protection against Source Inference Attacks in Federated Learning using Unary Encoding and Shuffling

Andreas Athanasiou*
andreas.athanasiou@inria.fr
INRIA and LIX, IPP
Palaiseau, France

Kangsoo Jung*
gangsoo.zeong@inria.fr
INRIA and LIX, IPP
Palaiseau, France

Catuscia Palamidessi
catuscia@lix.polytechnique.fr
INRIA and LIX, IPP
Palaiseau, France

Abstract

Federated Learning (FL) enables clients to train a joint model without disclosing their local data. Instead, they share their local model updates with a central server that moderates the process and creates a joint model. However, FL is susceptible to a series of privacy attacks. Recently, the source inference attack (SIA) has been proposed where an honest-but-curious central server tries to identify exactly which client owns a specific data record.

In this work, we propose a defense against SIAs by using a trusted shuffler, without compromising the accuracy of the joint model. We employ a combination of unary encoding with shuffling, which can effectively blend all clients' model updates, preventing the central server from inferring information about each client's model update separately. In order to address the increased communication cost of unary encoding we employ quantization. Our preliminary experiments show promising results; the proposed mechanism notably decreases the accuracy of SIAs without compromising the accuracy of the joint model.

CCS Concepts

• **Security and privacy**; • **Computing methodologies** → **Machine learning**;

Keywords

Federated Learning, Source Inference Attack, Unary Encoding, Shuffling

ACM Reference Format:
Andreas Athanasiou*, Kangsoo Jung*, and Catuscia Palamidessi. 2024. Poster: Protection against Source Inference Attacks in Federated Learning using Unary Encoding and Shuffling. In *Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security (CCS '24), October 14–18, 2024, Salt Lake City, UT, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3658644.3691411

CCS '24, October 14–18, 2024, Salt Lake City, UT, USA

ACM ISBN 979-8-4007-0636-3/24/10
https://doi.org/10.1145/3658644.3691411

1 Introduction

In FL [7], each client independently trains a model using their own data and then sends the model update to a central server. The server aggregates these model updates to create a new joint model, which is then distributed back to the clients. The process continues iteratively for multiple rounds, usually until the model converges. However, in a naive FL architecture, the central server can directly observe the clients' reported model updates. This may lead to various privacy attacks. For example, a colluded server could launch a *membership inference attack* (MIA) [4] in order to find whether a specific data point was included in *any* client's training dataset.

In this paper, we focus on *source inference attacks* (SIAs) [5], which aim to identify *exactly which* client owns a data point, in a setting where the central server is honest-but-curious. If successful, a SIA can lead to a severe violation of privacy; for instance, consider a scenario where several hospitals jointly build a medical model using patients' data to treat a disease. If an adversary identifies the hospital that owns a particular patient's record, and that hospital mostly treats COVID-19 patients, the attacker might infer that the patient suffers from COVID-19.

To the best of our knowledge, no effective defense to prevent SIAs has been proposed in the literature. A typical approach in privacy-preserving FL is to use *local differential privacy* (LDP) [8], where clients perturb their reported model updates by adding noise. However, this approach is not very suitable against a SIA, as it has been shown that the amount of noise necessary to prevent this kind of attacks would significantly deteriorate the accuracy of the joint model [5].

Contribution. In this work, our goal is to design a defense against SIAs that maintains high model accuracy. To this aim, we propose *Unary-Quant*; a mechanism involving a trusted shuffler which blends the clients' model updates before releasing them to the central server. The characteristic of this mechanism is that it does not require the addition of noise. Instead, it uses a unary encoding which, combined with shuffling, significantly reduces the amount of information available to the central server. To counter the high communication cost of unary encoding, Unary-Quant uses gradient quantization.

We experimentally evaluate the model accuracy of Unary-Quant on the MNIST dataset. The results show that almost no accuracy is

*Primary authors with equal contribution.

lost, i.e. the model accuracy is close to that of standard FL. Furthermore, we conduct experiments on SIAs. The results indicate that our proposed defense can significantly decrease the effectiveness of a SIA, in the sense that the accuracy of a source inference is reduced to nearly the level of a random guess.

2 Preliminaries

Federated Learning [7]. Federated learning aims to train a global ML model across N clients, each possessing its own local dataset $\mathcal{D}_i$. First, each client i updates the global model W using its local data $\mathcal{D}_i$ to generate an updated model w_i. Then, the central server aggregates the local updates from all clients to form the updated global model: $W \leftarrow \frac{1}{N}\sum_{i=1}^{N} w_i$ (FedAvg).

Quantization. In FL, to reduce the communication cost, *quantization* can be used to compress the model updates:

Definition 1 ([6]). *Let $h = (h_1, \ldots, h_\lambda)$ be the vector representation of a model parameter p. Let $h_{max} = \max_j(h_j)$ and $h_{min} = \min_j(h_j)$. The compressed version (unbiased estimator) of h, denoted by $\tilde{h}$, is: $\tilde{h} = h_{max}$ w.p. $\frac{h_j - h_{min}}{h_{max} - h_{min}}$ and $\tilde{h} = h_{min}$ w.p. $\frac{h_{max} - h_j}{h_{max} - h_{min}}$.*

Trusted Shuffling. In this work, we assume the presence of a trusted shuffler which has already been studied as a mean to protect privacy (for instance in the shuffle model of Differential Privacy (DP) [1]). Assuming the existence of a trusted shuffler can be considered as a smaller trust assumption compared to assuming that the central server is trusted since shuffling is a primitive operation that can be performed distributively (using MixNets or Multi-Party Computation) or using trusted hardware [2].

3 Protection against the SIA

First, let us clarify why just using standard (one-message) shuffling is *not* enough to efficiently protect against SIAs. While shuffling does initially break the link between the client and the model update, in FL the adversary may be able to re-identify each client. That is because the adversary might have some statistics over the clients' training datasets, which is often assumed in the literature of FL [4]. Hence he can use these statistics to remap the data owner and the reported model update, canceling the effect of the shuffler.

To overcome this obstacle and effectively blend all model updates, a more sophisticated approach to shuffling is necessary.

3.1 A first approach using Unary Encoding

To begin with, let us set aside the communication cost and discuss a simplified variant of Unary-Quant.

The core idea is, informally, that releasing a shuffled bit vector is privacy-wise equivalent to releasing its sum [2]. For example, take a bit vector of length 4 with 2 ones and 2 zeros. The statements: *"the sum of the vector is 2"* and *"the values of the vector (after shuffling) are $\{1, 0, 1, 0\}$"*, provide the adversary with the same amount of information. Observe that this applies only to bit vectors and not, for example, to integer vectors. However, in reality, most models involve parameters with values in $\mathbb{R}$, which are then typically bounded by clipping. In this work we assume w.l.o.g. that they are clipped in $[-1, 1]$ and introduce an encoding step (Algorithm 1) based on [2].

Algorithm 1: $E(x, r)$: Unary encoding of x [2]

Input : $x \in \mathbb{R}$ where $-1 \le x \le 1$, $r \in \mathbb{N}$
Output: $(b_1, \ldots, b_r) \in \{0, 1\}^r$
if $x = 0$ **then**
  Return $\{0\}^r$
$x' \leftarrow (1 + x)/2$;
Let $\mu \leftarrow \lceil x' \cdot r \rceil$ and $q \leftarrow x' \cdot r - \mu + 1$
for $j = 1, \ldots, r$ **do**

$$b_j = \begin{cases} 1 & \text{if } j < \mu \\ \text{Ber}(q) & \text{if } j = \mu \\ 0 & \text{if } j > \mu \end{cases}$$

Return $(b_1, \ldots, b_r)$

Now consider a mechanism as follows: every client trains their model and encodes every parameter p of the model update to a bit vector b of size r using $E(p, r)$. Then, every b is sent to the shuffler. Note that each message should also include some metadata describing what b represents (for example its layer number, if CNN is used). After all these bit vectors are shuffled, they are released to the central server which can aggregate them and form the joint model.

Observe that the released output of the shuffler completely prevents the adversary from distinguishing each local model and therefore performing a SIA. This is because only a shuffled vector of bits is available to the adversary. The only information from this vector that is useful to her is its sum, which only allows her to construct the joint aggregated model.

The Achilles' heel of this approach is its communication complexity. For example, if a CNN is used with n layers and each layer i has λ_i parameters, then each client has to send $r \cdot \sum_{i=1}^{n} \lambda_i$ bits. Despite the fact that this solution may still be applicable to the so-called *cross-silo* setting of FL, where each client typically has increased communication capabilities, we are about to explore in the following section a variant that decreases the cost while still offering sufficient protection.

3.2 Unary-Quant

Quantization can efficiently compress a model update, and since the result is an unbiased estimator of the initial value the impact on the model's accuracy is expected to be negligible.

The core idea of *Unary-Quant* is to use the expensive approach of Section 3.1 to transmit only the first k decimal places of each parameter of the model update; the rest can be transmitted through the cheaper (in terms of communication cost) quantization. In other words, we decompose each parameter p into two segments: p^a and p^b s.t. p^a contains the first k decimal places of the value and p^b contains the rest. Then unary encoding is used in the part p^a and quantization is used in the part p^b. The central server can combine the two parts, after they are shuffled, to form the joint model. Algorithm 2 provides an outline of Unary-Quant and Algorithm 3 shows how it is used in FL.

In essence, the adversary can only use the p^b segment to perform a SIA. Moreover, re-identifying each client only by her p^b is challenging and requires arguably strong assumptions (for example

the adversary knowing the clients' corresponding h_{min} and h_{max}). Note that in Algorithm 2, we applied 1-bit quantization, but it can be extended to n-bit quantization by dividing the range h_{min} and h_{max} into 2^n equal intervals [6].

Algorithm 2: Unary-Quant

Input : $x_j \in w_{t+1}^j, r \in \mathbb{N}, k \in \mathbb{N}$, where x_j has λ parameters and each parameter p is $-1 \leq p \leq 1$
Output: $U = (u_1, \ldots, u_\lambda), H = (h_1, \ldots, h_\lambda)$
$h_{max} := -1$; $h_{min} := 1$
for *each parameter* $p_i = p_1 \ldots p_\lambda$ *of* x_j **do**
  // Split p_i in parts
  $p_i^a := int(p_i) + \frac{\lfloor 10^k frac(p_i) \rfloor}{10^k}$
  $p_i^b := p_i - p_i^a$
  // Unary encoding of p_i^a
  $U_i \leftarrow E(p_i^a, r)$
  // Calculate h_{max} and h_{min}
  if $p_i^b > h_{max}$ **then**
    $h_{\max} = p_i^b$
  if $p_i^b < h_{min}$ **then**
    $h_{\min} = p_i^b$
// Quantization
for *each* $p_i^b = p_1^b, \ldots, p_\lambda^b$ **do**
  $H_i \leftarrow$ Quantization(p_i^b,$h_{\max}$,$h_{\min}$)
Return U, H

Algorithm 3: Federated Learning

Input : Number of rounds T, number of clients N
Output: Final global model w_R
Initialize global model w_0
for *each round* $t = 1, 2, \ldots, T$ **do**
  // Server-side
  Randomly select a subset of clients S_t of size $n \leq N$
  BroadcastGlobalModel(S_t,W_t)
  // Client-side
  for $j \in S_t$ *in parallel* **do**
    $w_{t+1}^j \leftarrow W_t - \eta \nabla \ell(W_t; \mathcal{D}_k)$
    U^j, H^j = Unary-Quant(w_{t+1}^j, r, k)
    Send U^j, H^j to shuffler
  // Shuffler-side
  Concatenate all U^j and H^j to a single vector U and H
  Send Shuffle(U) and Shuffle(H) to the server
  // Server-side
  $W_{t+1} \leftarrow$ **FedAvg**(U) + **FedAvg**(H)
Return : Final global model w_R

4 Preliminary Evaluation

In this section, we conduct a preliminary experiment to measure the effectiveness of Unary-Quant, in terms of both model accuracy and preventing SIAs, comparing it to the baseline of standard FL (i.e. without any defense mechanism). We use the MNIST dataset with 10 clients and use a Dirichlet distribution (setting its hyperparameter α to 0.1) to simulate the heterogeneity of the training data. We use a CNN model and the total number of model parameters is 421642.

First we measure the model loss using Unary-Quant with $k = 2$ and $k = 4$ while setting $r = 10^k$; Figure 1 shows that in both cases the model loss quickly approaches that of standard FL as the number of rounds increase. Table 1 shows that Unary-Quant achieves model accuracy nearly identical to standard FL while effectively protecting against SIAs: reducing their accuracy from 44.5% to 14.7%. Recall that the baseline of random guess is 10% (assumed to be uniform over all clients).

Method	Model Accuracy	SIA accuracy
Standard FL	98.8	44.5
Unary-Quant ($k = 3$, $r = 10^3$)	98.1	14.7

Table 1: Model and SIA accuracy after 15 rounds (percentage)

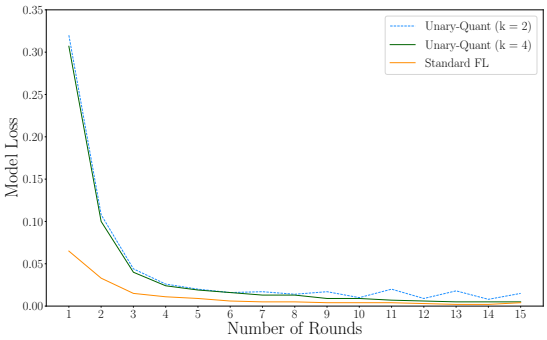

Figure 1: Model Loss

5 Discussion

The benefit of our approach is that it is primarily based on encoding, allowing for direct integration with other methods in FL that already use a trusted shuffler (e.g. the shuffle model of DP [3]). Our experiments indicate that Unary-Quant achieves model accuracy similar to that of standard FL while notably protecting against SIAs. More experiments should follow, measuring its effectiveness across multiple datasets with varying parameters (e.g. degree of heterogeneity, number of clients). Finally it is vital to explore additional gradient compression techniques as to further reduce the communication cost.

Acknowledgments

The work of Andreas Athanasiou was supported by the project CRYPTECS, funded by the ANR (project number ANR-20-CYAL-0006) and by the BMBF (project number 16KIS1439). The work of Kangsoo Jung was supported by the project ELSA, funded by the Horizon Europe Framework (project number 101070617). The work

of Catuscia Palamidessi was supported by the project HYPATIA, funded by the ERC (grant agreement number 835294).

References

[1] A. Bittau, Ú. Erlingsson, and P. Maniatis et al. 2017. Prochlo: Strong Privacy for Analytics in the Crowd. In *SOSP*. ACM.
[2] A. Cheu, A. Smith, J. Ullman, D. Zeber, and M. Zhilyaev. 2019. Distributed Differential Privacy via Shuffling. In *EUROCRYPT*. Springer.
[3] A. M. Girgis et al. 2021. Shuffled Model of Federated Learning: Privacy, Accuracy and Communication Trade-Offs. *IEEE J. Sel. Areas Inf. Theory* 2, 1 (2021), 464–478.
[4] Reza Shokri et al. 2017. Membership Inference Attacks Against Machine Learning Models. In *2017 IEEE SP*. 3–18. https://doi.org/10.1109/SP.2017.41
[5] H. Hu, Z. Salcic, L. Sun, G. Dobbie, and X. Zhang. 2021. Source Inference Attacks in Federated Learning. In *ICDM*. IEEE.
[6] J. Konečný and H. B. McMahan et al. 2016. Federated Learning: Strategies for Improving Communication Efficiency. *CoRR* abs/1610.05492 (2016).
[7] H. B. McMahan and E. et al. Moore. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *AIST*. PMLR.
[8] Y. Miao, R. Xie, X. Li, X. Liu, Z. Ma, and R. H. Deng. 2022. Compressed Federated Learning Based on Adaptive Local Differential Privacy. In *ACSAC*. ACM.