Exploiting Unintended Property Leakage in Blockchain-Assisted Federated Learning for Intelligent Edge Computing

Meng Shen, *Member, IEEE*, Huan Wang, Bin Zhang, Liehuang Zhu, *Member, IEEE*, Ke Xu, *Senior Member, IEEE*, Qi Li, *Senior Member, IEEE*, and Xiaojiang Du, *Fellow, IEEE*

***Abstract*—Federated learning (FL) serves as an enabling technology for intelligent edge computing, where high-quality machine learning (ML) models are collaboratively trained over large amounts of data generated by various Internet of Things devices while preserving data privacy. To further provide data confidentiality, computation auditability, and participant incentives, the blockchain framework has been incorporated into FL. However, it is an open question whether the model updates from participants in blockchain-assisted FL can disclose properties of the private data the participants are unintended to share. In this article, we propose a novel property inference attack that exploits the unintended property leakage in blockchain-assisted FL for intelligent edge computing. More specifically, we present an active attack to learn the property leakage from model updates of participants and to identify a set of participants with a certain property. We also design a dynamic participant selection strategy tailored to the setting of large-scale FL, which accelerates the selection process of target participants and improves attack accuracy. We evaluate the proposed attack through extensive experiments with publicly available data sets. The experimental results demonstrate that the proposed attack is effective and efficient in inferring various properties of training data, while maintaining the high quality of the main tasks in FL.**

***Index Terms*—Blockchain, edge computing, federated learning (FL), Internet of Things (IoT), property inference.**

Manuscript received April 30, 2020; revised July 19, 2020 and August 6, 2020; accepted September 19, 2020. Date of publication October 1, 2020; date of current version February 4, 2021. This work was supported in part by the Beijing Nova Program under Grant Z201100006820006; in part by NSFC Projects under Grant 61972039, Grant 61932016, and Grant 61872041; in part by the Beijing Natural Science Foundation under Grant 4192050; in part by the Zhejiang Lab Open Fund under Grant 2020AA3AB04; in part by the China National Funds for Distinguished Young Scientists under Grant 61825204; in part by the Beijing Outstanding Young Scientist Program under Grant BJJWZYJH01201910003011; in part by BNRist under Grant BNR2019RC01011; in part by the Science and Technology Planning Project of Guangdong Province under Grant LZC0023 and Grant LZC0024; and in part by the PCL Future Greater-Bay Area Network Facilities for Large-Scale Experiments and Applications under Grant LZC0019. *(Corresponding author: Bin Zhang.)*

Meng Shen is with the School of Cyberspace Security, Beijing Institute of Technology, Beijing 100081, China, and also with the Cyberspace Security Research Center, Peng Cheng Laboratory, Shenzhen 518066, China (e-mail: shenmeng@bit.edu.cn).

Huan Wang is with the School of Computer Science, Beijing Institute of Technology, Beijing 100081, China (e-mail: wanghuan6693@163.com).

Bin Zhang is with the Cyberspace Security Research Center, Peng Cheng Laboratory, Shenzhen 518000, China (e-mail: bin.zhang@pcl.ac.cn).

Liehuang Zhu is with the School of Cyberspace Security, Beijing Institute of Technology, Beijing 100081, China (e-mail: liehuangz@bit.edu.cn).

Ke Xu is with the Department of Computer Science and Technology and Beijing National Research Center for Information Science and Technology, Tsinghua University, Beijing 100084, China, and also with the Peng Cheng Laboratory, Shenzhen 518066, China (e-mail: xuke@tsinghua.edu.cn).

Qi Li is with the Institute for Network Sciences and Cyberspace, Tsinghua University, Beijing 100084, China (e-mail: qi.li@sz.tsinghua.edu.cn).

Xiaojiang Du is with the Department of Computer and Information Sciences, Temple University, Philadelphia, PA 19122 USA (e-mail: dxj@ieee.org).

Digital Object Identifier 10.1109/JIOT.2020.3028110

I. Introduction

In RECENT years, the rapid advancement of Internet of Things (IoT) results in a huge amount of data gathered from various IoT devices. The high-performance machine learning (ML) models require large amounts of data to perform data classification and prediction of future events [1]. To address the limitations of data privacy and network bandwidth, edge computing offloads computation resources and data to IoT devices. Federated learning (FL) and blockchain are emerging paradigms of distributed learning [2]–[4] that stores and processes data locally, which has been extended to the edge computing and used in various domains [5], [6]. With the advanced features such as anonymity and traceability, blockchain has emerged as a promising technology to provide distributed secure solutions in FL which provides a guaranteed collaborative scheme among untrusted participants and servers for efficient model training [4], [7]–[9].

Although the blockchain-assisted FL avoids the sharing of participants' data and guarantees the credibility and integrity of data, the shared model updates still reveal private information of participants' training data sets. An important question naturally arises: what can be disclosed about the participants' private data set from the model updates in blockchain-assisted FL?

Existing studies have investigated various privacy violations in the federated setting, such as membership inference attacks and property inference attacks. In the membership inference attack, an adversary can determine if an exact data record was used to train the model [10]–[12]. In the property inference attack, an attacker can infer properties of the training data that is uncorrelated with the main task, e.g., inferring the hair color or the race (as a property) of the images used to train a gender classifier [10], [12], [13].

In this article, we focus on the inference of *unintended properties* of the participates' training data, i.e., those properties that hold for the training data of certain subsets of the participants, but are not the *global* properties of the training data as a

whole. Different from prior work [12] that assumes an adversarial participant, we explore the unintended property leakage in FL with an adversarial *server*, as the server potentially has a greater attack capability, e.g., having a global view of the model updates of all participants, or even determining the participants involved at each iteration of training. This inference enables the server to extract valuable properties that participants are not intended to share and are independent of the features that characterize the main classifier.

In general, a practical property inference attack launched by a central server in FL has two goals. First, high inference accuracy is always desirable, which enables the adversary to correctly excavate as many private properties of participants' training data as possible. The second goal is to maintain the high utility of participants' training data to ensure the quality of the classifier of the main task. An ideal property inference attack should have a negligible impact on the main task.

Recently, the cryptography is widely used to protect the privacy of various data, such as graph [14] and text data [15]. In order to reduce the risk of information leakage from model updates in FL, several secure aggregation schemes and multiparty secure computing algorithms [16], [17] have been proposed to protect participants' local updates. Thus, we assume that an active adversary only obtains the aggregated model without knowing each participant's plaintext model updates. We also assume that the adversary has white-box knowledge about the federated model. Instead of inferring the properties of individual participants, the goal in our attack is to infer a subset of participants with the target properties that are of interest to the adversary, while maintaining the high quality of the classifier trained as the main task. It is a serious privacy risk for participants' sensitive data, especially in the field of healthcare and finance.

We design a participant selection strategy that can be actively launched by the server, which iteratively selects the participants whose training data is more likely with the target properties. Intuitively, the effectiveness and efficiency of the selection strategy are contradictory. Thus, we design a strategy with adjustable parameters, which enables the adversary to select appropriate parameters to meet their inference requirements.

We evaluate the effectiveness and efficiency of the proposed attack on several public data sets (i.e., CelebA [18], LFW [19], MNIST, and CASIA-WebFace [20]) with up to 1000 participants. The results show that the attack accuracy is mostly above 80%, even with 1000 participants. For example, when the main task is the gender classifier trained on the CASIA data set with 1000 participants, an adversary can infer 100 participants with "hair-color:black" with a considerable accuracy of 82.3%. Moreover, it almost preserves the same utility of the gender classifier, where the prediction accuracy slightly drops from 82.4% to 82.0% (LFW) and from 80.7% to 80.2% (MNIST).

Our contributions are summarized as follows.

1) We train a metaclassifier to exploit the property leakage in the training process of FL.
2) We design a selection strategy to iteratively infer a subset of participants with target properties from all participants in the large-scale FL.
3) We evaluate the proposed attacks on real-world data sets, which demonstrates that our attacks can efficiently infer the properties of participants' private data.

The remainder of this article is organized as follows. We introduce the related work and problem statement in Sections II and III. Next, we present the property attacks in Section IV. We describe the experimental data sets and evaluation results in Sections V and VI, respectively. Finally, we conclude this article in Section VII.

II. Related Work

FL provides a parallel scheme for participants to learn a collaborative model and achieves edge intelligence by learning from distributed data. The collaborative idea has been widely used in many fields [21], [22] and in the FL, it can reduce the privacy risk of directly sharing data [23]. Recently, blockchain is an emerging parallel that brings opportunities to the traditional information-centric networks [15], [24], [25] and has been widely used in FL for edge computing to provide data confidentially, computation auditability, device authentication for cross-domain industrial IoT and participant incentives [4], [7], [26], [27].

However, the model sharing does disclose the unintended leakage of the training data, which leads to various attacks against neural networks in the black-box setting or white-box setting. Here, we briefly summarize the inference attacks from two aspects: 1) membership inference attacks and 2) property inference attacks.

Membership Inference Attacks: Shokri *et al.* [28] proposed a membership attack to infer whether the target data records are the training data of the target model in a black-box setting. Yeom *et al.* [29] analyzed the influence of overfitting on the membership attacks. Long *et al.* [30] showed that the membership attacks were still effective even in the well-generalized learning models, because of the complexity and memorability of deep neural networks. Hayes *et al.* [31] presented the membership attacks against generative models combining the generative and discriminative models of generative adversarial networks (GANs) [32].

In collaborative learning, Nasr *et al.* [11] designed a membership attack model during the training phase in a white-box setting, including passive and active attackers based on the different adversary prior knowledge. Melis *et al.* [12] developed passive and active membership inference attacks to extract the unintended features from the model updates in the collaborative learning.

Property Inference Attacks: Model inversion attacks [33] were proposed to construct the inputs of a certain class. Ateniese *et al.* [34] built a metaclassifier to infer unexpected but useful statistical information of the training data set from ML classifiers. Ganju *et al.* [35] proposed the property inference attacks on fully connected neural networks to infer the global properties of the training data set and used the permutation invariant representations to simply the structures of networks.

In collaborative learning, the studies [10], [13] reconstructed the representation of a class based on the observing updates from participants using the GANs. Wang *et al.* [13]

TABLE I
MAIN NOTATIONS USED IN THIS ARTICLE

Notations	Paraphrases
$\mathbf{w}$	Deep neural network model
$L(\mathbf{w})$	Loss function of model $\mathbf{w}$
T	Required iterations of FT
$\mathbf{w}_t$	Global model at iteration t
$\mathbf{w}_t^i$	Local model of participant i at iteration t
N	The number of total participants in FL
$\mathcal{P}$	The target property that an adversary wants to infer
M	The number of participants with property $\mathcal{P}$
n_t	The number of selected participants at iteration t
K	The number of target participants with property $\mathcal{P}$
c	The size of participant sets
b	The number of screened participant sets

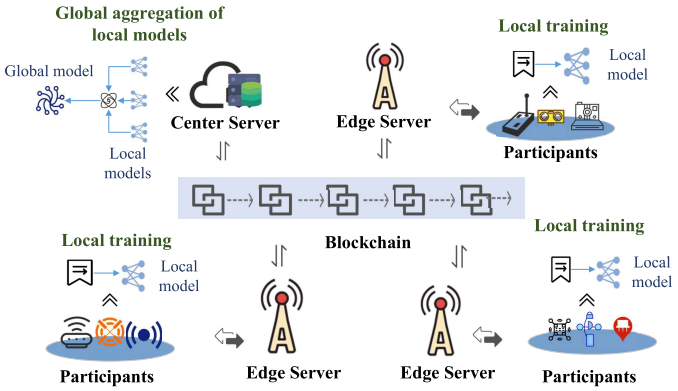

Fig. 1. Typical scheme of blockchain-assisted FL for intelligent edge computing.

proposed a novel GAN with a multitask discriminator to recover participants' specified private data from the server side. Melis *et al.* [12] designed a mini-batch property attack model to extract the unintended features in the FL [2] and collaborative learning with synchronized gradient updates [3] by active attacker and passive attacker.

In contrast, our property inference attacks could infer a set of participants with target properties by a server who can only obtain the aggregated global models, which achieves a more generic attack and has a negligible impact on the FL tasks.

III. PROBLEM STATEMENT

In this section, we present the background, threat model, and design goals of property inference attacks.

A. Background of Blockchain-Assisted Federated Learning

FL is an emerging paradigm of distributed learning with multiple participants. Participant i with data set D_i, at iteration[1] t ($t \in [0, T]$), trains the local model $\mathbf{w}_t^i$ that minimizes the local loss function $L_i(\mathbf{w}_t^i)$. The loss function is defined as

$$L_i(\mathbf{w}_t^i) = \frac{1}{|D_i|} \sum_{j \in D_i} l_j(\mathbf{w}_t^i) \tag{1}$$

where $l_j(\mathbf{w}_t^i)$ is the loss function of participant i on data sample (x_j, y_j). Define the global data set as $D = \bigcup_{i=1}^{N} D_i$, the objective of FL at iteration t is to train a global model $\mathbf{w}_t$ to minimize the global loss function $L(\mathbf{w}_t)$ as shown in

$$L(\mathbf{w}_t) = \frac{1}{|N|} \sum_{i \in N} \sum_{j \in D_i} \frac{L_i(\mathbf{w}_t^i)}{|D_i|} \tag{2}$$

where N is the number of participants and $|\cdot|$ denotes the size of sets. The main notations used in this paper are listed in Table I.

The distributed technologies, e.g., FL and blockchain, promote the development of edge computing. Edge computing provides the specific application scenarios for the combination of FL and blockchain which is widely used for enhancing security in areas such as, IoT [36] and vehicular networks [8]. The general process of the blockchain-assisted FL architecture [8], [9] typically consists of three phases: 1) nodes selection; 2) local training; and 3) global aggregation, as shown in Fig. 1.

1) *Nodes Selection:* At the beginning, the central server initializes the global model. Then, the central server selects the node to participate the model training.
2) *Local Training:* The local training is implemented with gradient descent. Upon receiving $\mathbf{w}_{t-1}$ in iteration t, each participant i updates the local model $\mathbf{w}_t^i$ on the local data D_i as illustrated in (1). Then, participant i sends the updated model $\mathbf{w}_t^i$ to the nearby edge server and uploads it to the blockchain for further verification and aggregation

$$\mathbf{w}_t^i = \mathbf{w}_{t-1} - \eta \nabla L_i(\mathbf{w}_{t-1}). \tag{3}$$

3) *Global Aggregation:* The aggregator (e.g., a centralized server) retrieves the updated models from the permissioned blockchain and aggregates local models $\mathbf{w}_t^i$ from participating nodes to a global model $\mathbf{w}_t$

$$\mathbf{w}_t = \frac{1}{n} \sum_{i=1}^{n} \mathbf{w}_t^i \tag{4}$$

where n is the number of nodes in iteration t.

B. Threat Model

In this article, we assume the server is an adversary. In the blockchain-assisted FL settings, the adversary is the edge server. It is worth mentioning that in the FL commonly used, our attack is still valid and the adversary is the central server in this setting. We also assume that the adversary has a white-box access to the structure of the federated model and federated algorithm. The assumption is also commonly used in [11] and [37].

In this setting, an adversary is unnecessary to strictly follow the procedures of the main task. Instead, he can take active actions (e.g., dynamically selecting the participants involved in each iteration) to improve the accuracy or efficiency of the property inference attack. We assume that secure aggregation schemes [17] are employed to protect the model updates of individual participants. Thus, the adversary has only access to the global model aggregated from the selected participants at

[1]The iteration is also called *round* in [2], [12], and [17].

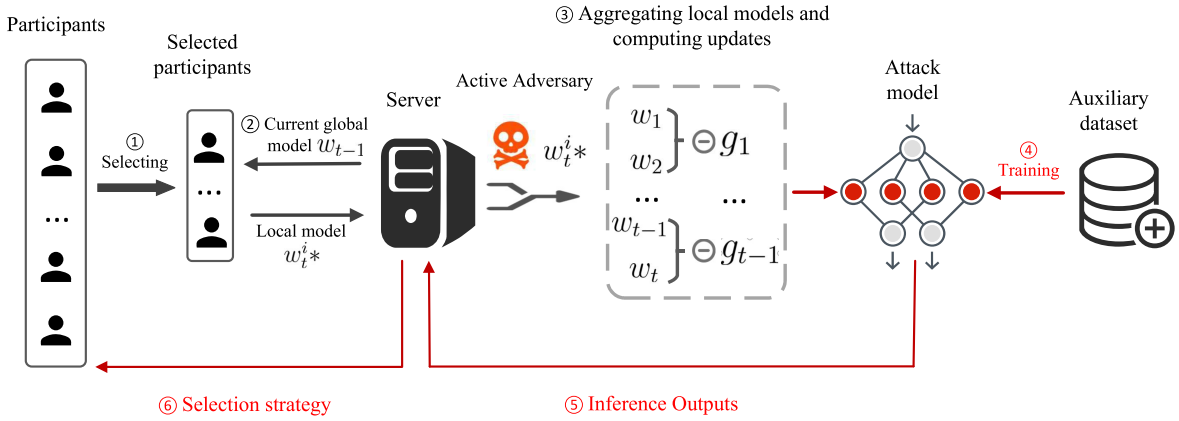

Fig. 2. Overview of the property inference attack. A honest server performs steps 1–3. The active adversary only obtains the global model w_t at the tth iteration and conducts steps 1–6 to perform the attack.

each iteration. In our attack, the goal turns to infer a certain number of participants that are most likely to have the desired properties in their training data.

In addition, to perform the property inference attacks, the adversary needs an auxiliary data set that has a similar distribution with the participants' private data. That is because the data sets with similar distribution have similar gradients of deep neural networks [11], [28], [35]. The auxiliary data record is correctly labeled with the property an adversary wants to infer, as well as the label of the main task.

If the adversary does not have access to similar training data set (e.g., without publicly available data sets), he can compromise one of the participants to obtain the auxiliary data set, as evidenced in FL with large-scale participants [38].

C. Design Goals

An ideal property inference attack should allow an adversary to extract the unintended property leakage of participants' private data from the model updates, while preserving the quality of the main task classifier. Therefore, the design goals can be described in the following aspects.

1) *High Accuracy:* For an active adversary, the goal is to infer a certain number of participants with the target property. The attack accuracy represents the ratio of correct inferences, i.e., the percentage of participants with the target property in the inferred participants.
2) *High Efficiency:* In the property inference attack, the attack efficiency is the required iterations of this attack during the FL process. We choose the iterations of federated training rather than running time of the attack for two reasons. First, the iterations represent the communications between adversary and participants, the fewer communications, the more effective of attack. Another reason is that the time cost of each iteration depends on the performance of different machines.
3) *Low Overhead:* In the active attack scenario, the adversary actively selects certain participants in the training process of FL, which can have an impact on the accuracy of the main tasks. Thus, we use overhead to evaluate the impact of the proposed attacks on the main tasks of FL.

IV. Property Inference Attacks

In this section, we describe the attack model used to extract the property leakage of private data sets of participants from model updates.

A. Property Inference Attack Model

We present a high-level overview of the property inference attacks in Fig. 2. As mentioned above, for the adversary, he takes steps 1–4 to train the attack model on the auxiliary data set. Then, he feeds the global updates $\mathbf{w}_t$ to the attack model and uses the selection strategy (step 6) to select participants with the target property based on the outputs of the attack model.

Define $\mathcal{P}$ as the target property the adversary aims to infer, and $\mathcal{H}$ as a metaclassifier to determine whether the target participant's private data has the property $\mathcal{P}$ or not.

Following research [11], [35], to train $\mathcal{H}$, the adversary has an auxiliary data set denoted by D_a, which has similar distribution with the training data for the federated model. $D_a = \{D_a^{\mathcal{P}}, D_a^{\overline{\mathcal{P}}}\}$, where $D_a^{\mathcal{P}}$ and $D_a^{\overline{\mathcal{P}}}$ are the auxiliary data sets with and without the property $\mathcal{P}$, respectively. The property inference attack mainly consists of three phases: 1) training data generation; 2) attack model training; and 3) property inference, as illustrated in Algorithm 1.

Generating Training Data Set of Attack Models: To train the attack model $\mathcal{H}$, an adversary first generates the training data set of $\mathcal{H}$. $\mathcal{H}$ is used to learn the differences of gradients of model updated on the private data sets with and without $\mathcal{P}$. Therefore, the server mimics the behavior of the target model and divides the auxiliary data set into N pieces. Then, at each iteration t, the adversary trains the federated model $\mathbf{w}_t$ on N auxiliary data sets and obtains gradients on auxiliary data sets $D_a^{\mathcal{P}_i}$ $(i \in [1, x])$ and $D_a^{\overline{\mathcal{P}}_j}$ $(j \in [1, y])$ after obtaining the global model $\mathbf{w}_{t-1}$, just like the participants in the FL. x is the number of divided pieces with the property $\mathcal{P}$ and y is the number of divided pieces without $\mathcal{P}$, where $D_a^{P} = \sum_{i=1}^{x} D_a^{P_i}$, $D_a^{\bar{P}} = \sum_{j=1}^{y} D_a^{\bar{P}_j}$. The update rule is defined in

$$g_a^{\mathcal{P}} = \nabla L\left(D_a^{\mathcal{P}_i}, \mathbf{w}_{t-1}\right)$$

Algorithm 1 Inference Attack Model

Input: auxiliary dataset $D_a = \{D_a^{\mathcal{P}}, D_a^{\overline{\mathcal{P}}}\}$, model $\mathbf{w}_t$
Output: Attack model $\mathcal{H}$

Server executes: Initialize global model $\mathbf{w}_0$, T;
for $t = 0$ to T **do**
  /* Generating training datasets of $\mathcal{H}$ */
  $D_c^{\mathcal{P}} = \phi, D_c^{\overline{\mathcal{P}}} = \phi$
  for $k = 1$ to K **do**
    for each $D_a^{\mathcal{P}_i}$ $(i \in [1, x])$, $D_a^{\overline{\mathcal{P}}_j}$ $(j \in [1, y])$
      $g_a^{\mathcal{P}} = \nabla L(D_a^{\mathcal{P}_i}, \mathbf{w}_{t-1}), g_a^{\overline{\mathcal{P}}} = \nabla L(D_a^{\overline{\mathcal{P}}_j}, \mathbf{w}_{t-1})$ **do**
      $D_c^{\mathcal{P}} \leftarrow D_c^{\mathcal{P}} \cup (g_a^{\mathcal{P}}, \mathcal{P}), D_c^{\overline{\mathcal{P}}} \leftarrow D_c^{\overline{\mathcal{P}}} \cup (g_a^{\overline{\mathcal{P}}}, \overline{\mathcal{P}})$
    end for
  end for
  Train attack model $\mathcal{H}$, given $D_c^{\mathcal{P}}, D_c^{\overline{\mathcal{P}}}$
  $S_t = Select(n)$ /* Participant selection strategy */
  for each $i \in S_t$ **do**
    $\mathbf{w}_t^i$ = **ClientUpdate**$(\mathbf{w}_{t-1})$
  end for
  $\mathbf{w}_t = \sum_{k=1}^{n} \frac{1}{n} \mathbf{w}_t^k$
  /* Prediction phase of attack model */
end for

$$g_a^{\overline{\mathcal{P}}} = \nabla L\left(D_a^{\overline{\mathcal{P}}_j}, \mathbf{w}_{t-1}\right) \tag{5}$$

where $L(D, \mathbf{w})$ is described in (1).

To simplify the input features and attack model, we extract only the parameters of the last fully connected layer. The reason is that the last layer of the neural network leaks more information about the training data set [11]. Hereafter, $\mathbf{g}$ represents the gradients of the last layer.

Training Phase of the Attack Model: The adversary builds the training data of the attack model $\mathcal{H}$ by labeling the gradients $g_a^{\mathcal{P}}$ and $g_a^{\overline{\mathcal{P}}}$ as $\mathcal{P}$ and $\overline{\mathcal{P}}$, respectively.

Prediction Phase of the Attack Model: The adversary aggregates the global updates and uses it as the input feature of $\mathcal{H}$. Then, the attack model $\mathcal{H}$ outputs a score in [0, 1]: the closer the score is to 1, the more likely the updates have property $\mathcal{P}$, and *vice versa*. After that, the adversary leverages the participant selection strategy to iteratively select participants with the property $\mathcal{P}$ based on the predictions of $\mathcal{H}$.

B. Naive Selection Strategy

To select a set of participants with the target property, we first consider a naive selection strategy (NSS). At each iteration of FL, the server randomly selects n participants involved in model training. If we aim to infer K participants with the target property, an NSS is to select $K(K = n)$ participants at each iteration, and infer the set of participants with the largest probability of having the target property.

The NSS can be described as follows.

1) *Partitioning Participants:* The adversary divides N participants into multiple sets, each of which contains K participants. Thus, the number of selected participants n at each iteration equals K.
2) *Screening Out Participants:* At each iteration, the server selects a set of K participants and predicts the probability that the global update of the selected participants has a certain property. After selecting all participant sets, the adversary obtains the corresponding predictions of $\mathcal{H}$. Then, he screens out the last one sets of participants $(b \geq 1)$ based on the rank of predictions, as the prediction represents the probability that the set of participants has the property. Thus, the adversary selects the participants may have the property $\mathcal{P}$. Repeat the above steps until only one set of participants is left.

Algorithm 2 DSS

Input: Attack model $\mathcal{H}$ and global model $\mathbf{w}_t$
Output: The set of participants $\mathcal{S}_{\mathcal{P}}$, $|\mathcal{S}_{\mathcal{P}}| = K$

Initialize $N_0 = N$, $\mathcal{S}_{\mathcal{P}} = \emptyset$
for $r = 0$ to R **do**
  Divide N_r participants into c sets, $S_r = \{S_r^1, S_r^2, \ldots, S_r^c\}$
  for each $S_r^i \in S_r$ **do**
    $\mathbf{w}_t = \sum_{k=1}^{|S_r^i|} \frac{1}{|S_r^i|} \mathbf{w}_t^k$, $g(S_r^i)) = \mathbf{w}_t - \mathbf{w}_{t-1}$
    $Pre(S_r^i) \leftarrow (\mathcal{H}, g(S_r^i))$
  end for
  Filter out the b sets S_r^x with smallest prediction, $x \in [1, c]$
  $N_r = N_r - b \times \lfloor N_r/c \rfloor$, $S_r = S_r - \{S_r^{x_1}, S_r^{x_2}, \ldots, S_r^{x_b}\}$
end for
$\mathcal{S}_{\mathcal{P}} = S_r$

There are two main problems in NSS. First, the accuracy of $\mathcal{H}$ will drop as the number of target participants K increases. This is because local updates are averaged, making it more difficult to extract information from the updates. Second, given the number of participants N, the required iterations of this attack are only affected by the number of target participants K. For instance, if $N = 100$ and $K = 10$, the number of participant sets is $\lfloor N/K \rfloor = 10$, and the adversary needs ten iterations to traverse all participant sets. This limits the flexibility in participant selection and thus reduces the attack efficiency.

C. Dynamic Selection Strategy

To select a given number of participants more efficiently, while achieving high accuracy of the attack model, we propose a dynamic selection strategy (DSS) to adaptively select participants at each iteration.

The basic idea of DSS is illustrated in Fig. 3. We use two parameters c and b to dynamically change the number of selected participants and the number of screened participant sets. At each iteration, the number of selected participants is set as $n = \lfloor N/c \rfloor$, where c represents the proportion of the selected participants among all participants. Also, the number of screened participants equal $b \times n$, where n is the size of the selected participants set and b is the number of screened participant sets.

Algorithm 2 exhibits the process of DSS. We first divide the participants into c sets randomly, and iteratively screen out b sets of participants with the minimum probability of owning the property $\mathcal{P}$, until the required number of participants are left. However, it is important to note that these c sets do not overlap. R is the round of selection strategy, and we initialize $N_1 = N$ and $M_1 = M$, where M denotes the number of participants with property $\mathcal{P}$. At each round r $(r \in [1, R])$, we perform the selection strategy as shown in Fig. 3.

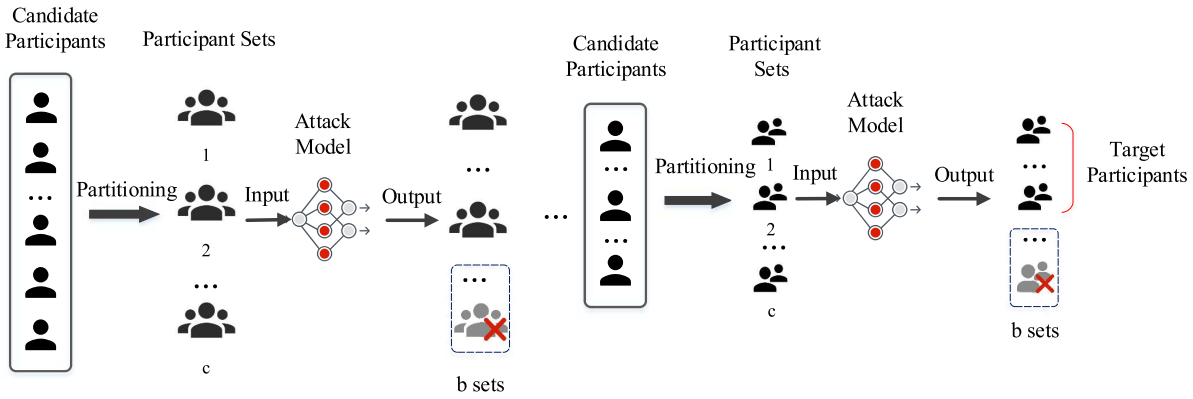

Fig. 3. DSS used in the active attack scenario.

1) *Partitioning Participants:* We divide the N_r candidate participants into c sets, each participant set contains $\lfloor N_r/c \rfloor$ participants and the number of selected participants $n = \lfloor N_r/c \rfloor$.
2) *Screening Out participants:* At each iteration, the server selects a set of participants S_r^i $(i \in [1, c])$ and predicts the probability that the global update from S_r^i has the property $\mathcal{P}$ using the metaclassifier $\mathcal{H}$. After c iteration, we obtain the corresponding predictions of $\mathcal{H}$. Then, we screen out the last b sets of participants S_r^j $(j \leq c)$ based on the rank of predictions, since the predictions represent the probability that the set of participants has the property $\mathcal{P}$. Thus, we screen out the participants most likely without the property.

After steps 1) and 2), the number of candidate participants is $N_{r+1} = N_r - b \times \lfloor N_r/c \rfloor$, and remaining participants with the property $\mathcal{P}$ satisfies Definition 1. Repeat the above steps until the number of candidate participants satisfies $N_R \approx K$, where K is the number of participants with the property $\mathcal{P}$ we aim to infer. It should be noted that some participants who are not selected in this iteration (less than $\lfloor N_r/c \rfloor$) will not be screened out.

Definition 1 (Candidate Participants): Given the number of candidate participants N, the number of participants M with the property $\mathcal{P}$, the number of selected participants n at each iteration of FL, the number of sets of participants c, a screen parameter b, the remaining candidate participants N_r, and the remaining participants with the property $\mathcal{P}, M_r$ following the rules:

$$N_r = N\left(1 - \frac{b}{c}\right)^r \tag{6}$$

$$M_r = M - \sum_{j=1}^{r}\sum_{i=1}^{b} x_j^i \tag{7}$$

where $x_j = \sum x_j^i$ $(i \in [1, b])$ is the number of screened participants with property $\mathcal{P}$ at the jth round of DSS.

D. Effectiveness Analysis of DSS

In this section, we formalize the effectiveness and efficiency of the above selection strategy and analyze the impact of the number of participants sets c and screened participant sets b.

We use $\mathcal{S}$ to represent the above selection strategy, $\mathcal{A}$ and $\mathcal{B}$ represent the accuracy and efficiency of this attack. Attack accuracy $\mathcal{A}$ is the proportion of participants with the property $\mathcal{P}$ among the inferred participants.

We use the required iterations of this attack to formalize its efficiency, because the time cost of each iteration in the FL depends on the performance of machines.

The selected participants with the property $\mathcal{P}$ at each round subject to the hypergeometric distribution defined as follows.

Definition 2 (Hypergeometric Distribution): Given the above variables N, M, and n. The probability of the number of participants X with property $\mathcal{P}$ in n selected participants satisfies the hypergeometric distribution $X \sim H(N, M, n)$. It follows the rule, and the mathematical expectation of X is $E(x) = (nM/N)$:

$$P(X = x) = \frac{C_M^x C_{N-M}^{n-x}}{C_N^n}. \tag{8}$$

Given the above selection strategy $\mathcal{S}$, the impacts of c and b on attack accuracy are defined as follows.

Definition 3 (Selection Strategy): Given a hypergeometric distribution $X \sim H(N, M, n)$, the parameters of b and c, the target number of participants K with the property $\mathcal{P}$. We define the selection $\mathcal{S}$ to maximize the accuracy of our objective by solving the following problems:

$$\arg\min_{c,b} \sum_{r=1}^{R}\sum_{i=1}^{b} P\left(X = x_r^i\right), X \sim H(N_r, M_r, n_r). \tag{9}$$

The efficiency of our attack is iterations in Algorithm 1, and is calculated as follows.

Definition 4 (Iterations Required): Given the selection strategy $\mathcal{S}$, the parameters of c, b, the round required R of the selection strategy, the iterations required is $r \times c$, thus, the efficiency objective aim to solving the following problem:

$$\arg\min_{c,b}\left(c \times log\left(1 - \frac{b}{c}\right)^{\frac{K}{N}}\right). \tag{10}$$

From (9) and (10), we can see that the effectiveness and efficiency of DSS are related to the parameters b and c. The larger the number of participant sets c, the higher the accuracy of the attack and the more iterations needed. The smaller the

screen value b, the higher the accuracy of the attack and the more iterations needed.

We demonstrate the efficiency and effectiveness of DSS in Section VI.

V. Data Sets and Model Architectures

In this section, we describe the data sets and the architecture of the target models and the inference attack models.

A. Data Sets

Labeled Faces in the Wild (LFW): It is a standard data set designed for face recognition, and contains 13 233 face images of 5749 individuals. Each image has multiple labels, such as race, age, gender, hair color, and eyewear. As shown in Table II, for LFW_1, the main task is the gender classifier and the target property is "race:black"; for LFW_2, the main task is the race classifier and the target property is male. Each participant in FL has 32 images (i.e., the batch size).

CelebFaces Attributes (CelebA): It contains 202 599 face images for 10 177 celebrities. Each image labels 40 binary attribute annotations, such as race, smile, age, and gender. The main tasks of FL on CelebA are gender and smile classifiers, and the target properties are race, smile, and black hair, which are denoted by CelebA_1, CelebA_2, CelebA_3, and CelebA_4, respectively. Each participant in FL has 64 images, composing a total of 128 000 images.

CASIA-WebFace: It contains more than 400 000 face images of 10 575 individuals. For CASIA_1, the main task is gender classifier and the target property is the black race; and for CASIA_2, the main task is race classifier and the target property is male. Each participant in FL has 64 images.

MNIST: It contains 70 000 handwritten digits. We select a subset of the original data set, named MNIST_1. The main task is to recognize the digit shown in the image, and we want to infer whether the target model was trained using noisy images [35]. We create the noisy images by adding a random brightness jitter to each image as suggested in [35]. Each participant in FL has 32 images.

We divide the data sets randomly into two parts for adversary and participants, as shown in Table III. It should be noted that there is no overlapping between the auxiliary data set of the adversary and the training data sets of participants. The data set sizes for all participants are the same on the same data set. Half of the participants have the training data sets with property $\mathcal{P}$, and another half of the participants have the training data sets without property $\mathcal{P}$. Note that each participant's private data only has pure properties, e.g., either black hair or nonblack hair.

B. Model Architecture Description

Target Models: For the face classification task, we use a three-layer CNN classifier with 32, 64, and 128 filters at each layer, a kernel size of (3, 3), followed by two fully connected layers of size 256 and 2. We use the FaceNet [39] to align all face images to 160×160 pixels. On MNIST, we use a two-layer CNN classifier with 32, 64 filters, followed by two fully connected layers of size 128 and 10. We use the ReLU as the activation function and SGD learning algorithm for all models. The learning rate is 0.00001. The batch sizes are 32 on LFW and MNIST and 64 on the rest data sets.

TABLE II
Data Sets and Tasks in Our Experiments. We Use the Pearson's Coefficient to Show the Relevance of the Target Properties to the Main Task Labels

Datasets	#Records	Main task	Target property	Corr
LFW_1	12.8K	Gender	Race (black)	0.084
LFW_2	16.6K	Race	Gender (male)	0.084
CelebA_1	128K	Gender	Smile	-0.139
CelebA_2	128K	Smile	Gender (male)	-0.139
CelebA_3	128K	Gender	Black Hair	0.114
CelebA_4	128K	Smile	Attractive	0.054
CASIA_1	102.4K	Gender	Race (black)	-
CASIA_2	102.4K	Race	Gender (male)	-
MNIST_1	57.6K	Recognition	Noisy images	-

TABLE III
Sizes of Data Sets Used in Main Tasks and Inference Tasks

Datasets	Target model		Attack model	
	Dataset with p	Dataset without p	Dataset with p	Dataset without p
LFW_1	3,200	3,200	3,200	3,200
LFW_2	3,200	3,200	9,600	640
CASIA_1	32,000	32,000	6,400	32,000
CASIA_2	25,600	12,800	32,000	32,000
$\text{CelebA}_{1,2,3,4}$	32,000	32,000	32,000	32,000
MNIST_1	16,000	16,000	12,800	12,800

Attack Model: We flatten the input features into a 1-D vector and use a CNN with 100 kernels of size (1, 100) to extract the input features. The max-pooling size is (1, 2). Two fully connected layers are of size 128 and 64. We use the ReLU as the activation function and Adam optimizer for all inference attack models. The learning rate is 0.001 and the output of attack models is a softmax layer.

VI. Performance Evaluation

In this section, we evaluate the performance of the proposed attacks.

A. Evaluation of Property Inference Attack

Recall that the goal of the adversary is to infer a certain number of participants with the target properties. We set the number of target participants $K = (N/10)$, unless otherwise noted. Table II shows an overview of our experiments. We use the Pearson's coefficient to show the relevance of the target properties to the main task labels. The Pearson's coefficients are missing in CASIA and MNIST data sets, because there are no data labels in CASIA and MNIST, only the images.

Attack Accuracy: To evaluate the accuracy of the active attack, in each experiment in Table II, we randomly initialize model parameters to repeat the inference attack 100 times and obtain the averaged attack accuracy. We also vary the participant number N from 100 to 1000 to evaluate its performance with different participant populations.

TABLE IV
ACCURACY AND ITERATION REQUIRED FOR ACTIVE PROPERTY INFERENCE ATTACKS ($K = [N/10]$)

Experiments	N=100		N=200		N=400		N=600		N=800		N=1,000	
	Acc.	Iterations	Acc.	Iterations	Acc.	Iterations	Acc.	Iterations	Acc.	Iterations	Acc.	Iterations
LFW_1	90.7%	60	87.6%	96	-	-	-	-	-	-	-	-
LFW_2	89.2%	60	84.5%	96	-	-	-	-	-	-	-	-
$CelebA_1$	77.3%	60	77.6%	96	72.2%	148	71.0%	187	70.2%	260	71.3%	402
$CelebA_2$	72.1%	34	76.0%	96	72.6%	209	70.8%	236	71.2%	385	70.2%	462
$CelebA_3$	76.2%	60	73.1%	57	74.8%	148	69.5%	297	71.2%	228	70.1%	298
$CelebA_4$	70.0%	40	70.1%	96	71.5%	185	69.5%	297	68.8%	385	66.5%	402
$CASIA_1$	91.3%	34	96.0%	96	90.3%	154	86.2%	150	80.7%	139	82.3%	221
$CASIA_2$	93.6%	60	91.4%	96	83.7%	105	83.0%	150	83.6%	228	82.6%	298
$MNIST_1$	89.0%	60	87.5%	96	81.3%	105	81.2%	150	81.6%	228	81.5%	307

TABLE V
COMPARISON OF ATTACK ACCURACY AND EFFICIENCY BETWEEN DSS AND NSS ($N = 100$ AND $b = 2$)

Experiments	DSS		NSS	
	Accuracy	Iterations	Accuracy	Iterations
LFW_1	83.9%	34	78.4%	54
LFW_2	84.8%	34	72.5%	54
$CelebA_1$	73.8%	41	60.3%	54
$CelebA_2$	73.4%	40	60.6%	54
$CelebA_3$	75.0%	34	64.8%	54
$CASIA_1$	90.9%	34	75.4%	54
$CASIA_2$	87.2%	34	72.0%	54
$MNIST_1$	81.9%	39	68.2%	54

TABLE VI
ACCURACY OF THE MAIN TASKS WITH OR WITHOUT ATTACKS FOR 240 ITERATIONS ($N = 100$ AND $b = 2$)

Experiments	Attack Accuracy	Accuracy of Main Tasks	
		Attack	Attack-free
LFW_1	83.9%	81.1%	81.3%
LFW_2	84.8%	88.9%	89%
$CelebA_1$	73.8%	95.3%	95.4%
$CelebA_2$	73.4%	90.6%	90.7%
$CelebA_3$	75.0%	85.2%	85.5%
$CASIA_1$	90.9%	85.2%	86%
$CASIA_2$	87.2%	86.4%	86.4%
$MNIST_1$	81.9%	80.0%	80.2%

Table IV shows the accuracy with different data sets. The attack accuracy decreases as the number of participants N increases, but remains above 70% even with 1000 participants.

To verify the effectiveness of the active selection strategy, we compare DSS with NSS. Table V shows the accuracy of the active attacks using NSS or DSS with $c = 7$ and $b = 2$. The number of participants N equals 100 in all experiments. The results show that DSS helps increase the attack accuracy while significantly reducing the iterations required.

Attack Efficiency: We use the required iterations during the inference attack to evaluate the effectiveness of active attacks, as illustrated in (10). Table IV shows the required iterations in different data sets with varying participant numbers.

Attack Overhead: The only difference of the active attack from the passive attack lies in that the adversary can actively select participants invovled in the training of main tasks. Table VI shows the impact of the active attack on the accuracy of the main tasks with different data sets. We can see that the accuracy of the main tasks only slightly drops when the active attacks are launched.

Impact of Target Participants on Inference Attack: Table VII shows attack accuracy for a different number of target participants. As expected, an increasing number of target participants decreases the accuracy of the property inference attack. It should be noted that in all experiments, a half of the participants have the target property $\mathcal{P}$, which means that $(N/2)$ is the upper bound of K.

Impact of Training Iteration on Inference Attack: Fig. 4 shows the attack accuracy performed in different iterations of main tasks. As expected, the training iteration has a marginal impact on the attack accuracy, and increasing the training iterations weakens the accuracy of the property inference attack. When the models of main tasks have been fitted, the update gradients of the main task models will be less obvious. From the above, the more training iterations, the higher the accuracy of the main task models, and the less significant the update gradients.

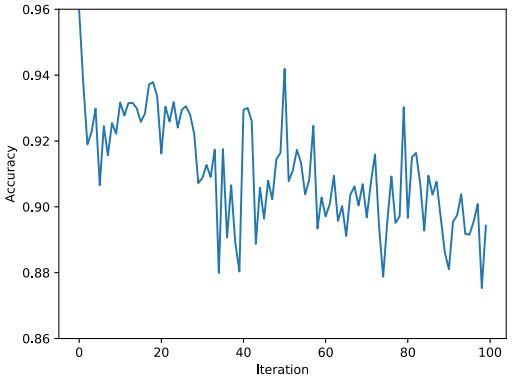

Fig. 4. Accuracy of inference attack in various iterations on LFW_1 ($N = 100$ and $K = 10$).

Impact of Distributions of Participants' Data Sets on Inference Attack: We evaluate the attack accuracy for various distributions of the participant's private data in Table VIII. If the size of the participant's data with property $\mathcal{P}$ is larger than those with $\bar{\mathcal{P}}$, we label the model updates as $\mathcal{P}$ and *vice versa*. As expected, the closer the size of data with $\mathcal{P}$ and $\bar{\mathcal{P}}$, the worse our attack performance.

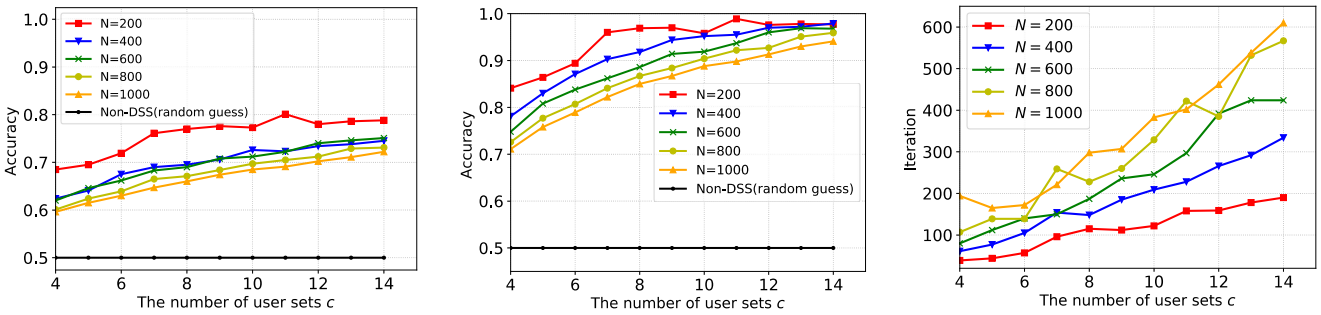

Fig. 5. Attack accuracy and efficiency with varying N and c ($K = (N/10)$, $M = (N/2)$, and $b = 2$). (a) Accuracy in CelebA$_1$. (b) Accuracy in CASIA$_2$. (c) Efficiency in CelebA$_1$ and CASIA$_2$.

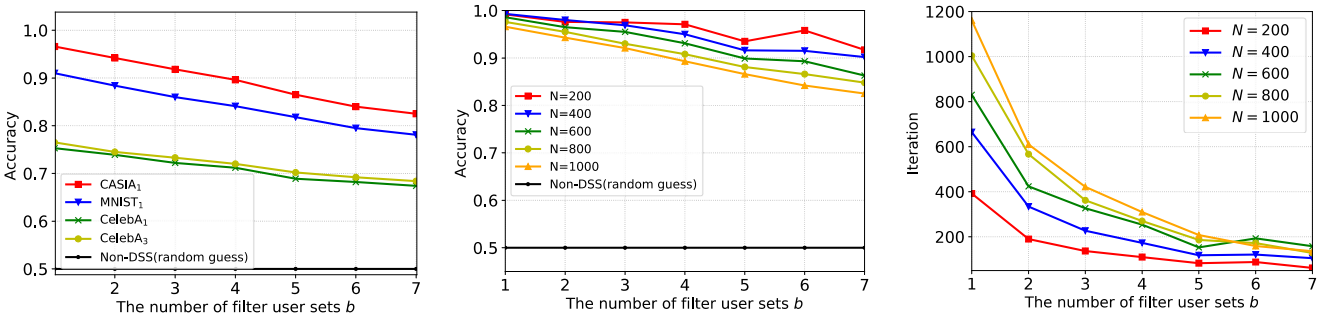

Fig. 6. Attack accuracy and efficiency with varying b ($c = 14$). (a) Accuracy with $N = 1000$. (b) Accuracy in CASIA$_1$. (c) Efficiency in CASIA$_1$.

TABLE VII
ATTACK ACCURACY AT INFERRING VARIOUS SIZE OF TARGET PARTICIPANTS IN CASIA$_1$ ($c = 9$ AND $b = 1$)

Participants (N)	Attack accuracy with different (K)			
	$1/10N$	$1/5N$	$1/3N$	$1/2N$
100	89.0%	89.9%	81.3%	70.7%
200	91.2%	87.9%	79.5%	71.1%
400	89.7%	84.6%	76.3%	68.6%
800	85.4%	79.7%	72.1%	65.2%
1000	84.1%	78.1%	70.5%	64.3%

TABLE VIII
ATTACK ACCURACY FOR VARIOUS DISTRIBUTIONS OF PARTICIPANT'S PRIVATE DATA. THE SIZE OF PARTICIPANT'S DATA IS 64. THE RATIO OF THE DATA WITH $\mathcal{P}$ TO THE DATA WITH $\bar{\mathcal{P}}$ ARE 7/1, 4/1, 3/1, AND 2/1 ($N = 100$, $c = 7$, AND $b = 3$)

Ratios of Pro. Sizes and Non-Pro. Sizes	Pro. Sizes	Non-Pro. Sizes	Attack Acc.
1/0	64	0	86.3%
7/1	56	8	84.8%
4/1	51	13	83.6%
3/1	48	16	71.9%
2/1	43	21	71.3%

B. Impact of Selection Strategy on Inference Attack

To select the suitable parameters of DSS, we analyze the experimental results with varying the size of selected participants c, the number of screened participant sets b, and the number of participants N.

Impact of the Number of Participants N: Fig. 5(a) and (b) demonstrates the impact of the number of participants N on CASIA-WebFace and CelebA data sets. From Fig. 5, the attack accuracy of $N = 200$ is obviously higher than the accuracy of $N = 1000$ with the same parameters c and b. As expected, the larger N is, the more iterations are required, the lower efficiency and accuracy the attack will get.

Impact of the Size of Selected Participants c: Fig. 5 shows the accuracy and efficiency of this attack on CelebA and CASIA data sets with different c. We set the beginning value of c as 4 rather than 1. The reason behind this is twofold. The first one is that c represents the proportion of selected participants among all participants which means it has a lower bound. The second reason is that when c is smaller, the number of selected participants $n = \lfloor N/c \rfloor$ would be larger, and the accuracy of the attack model $\mathcal{H}$ would be lower. When c increases, the number of iterations increases, and the efficiency of this attack would decrease as well.

Impact of Screened Participant Sets b: As shown in Fig. 6(a) and (b), the attack accuracy is negatively correlated with b. The attack efficiency is positively correlated with b. When the value of b increases, the required iterations decrease and the efficiency of this attack would increase [see Fig. 6(c)].

From these experiments in several real-world data sets, we verify the impact of parameters N, c, and b on the effectiveness and efficiency of the attacks and the selection strategy we proposed.

C. Limitations of Attacks

Attack Model: The attack model we proposed is used to predict whether a model update has a certain property. The adversary generates metatraining data to train the attack model based on the global model and auxiliary data set. When the number of required iterations is large, the time cost of training the attack model is high.

Number of Selected Participant: The output of the attack model is the probability that the model update has a target property. When the number of selected participants increases, the accuracy of the attack model will decrease. In order to ensure the accuracy of the attack model and iteratively inferring the participants with target property, the number of selected participants is limited.

VII. Conclusion

In this article, we proposed a novel attack to exploit the unintended properties leakage from model updates in blockchain-assisted FL for intelligent edge computing. This attack is practicable and enables a server to infer a set of participants with target properties, which is a risk to the sensitive data of IoT devices. We evaluated the proposed attacks on real-world data sets and demonstrated that the proposed attacks are effective and efficient in inferring various properties of training data while having a negligible impact on the main tasks of FL. Our attacks suggest that there are a number of privacy risks even if the local updates of participants are encrypted and servers observe only the aggregated global updates. In future work, we will improve the attack efficiency and explore better defenses to protect sensitive information of participants in the training process of blockchain-assisted FL.

References

[1] S. Wang *et al.*, "Adaptive federated learning in resource constrained edge computing systems," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 6, pp. 1205–1221, Jun. 2019.

[2] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. 20th Int. Conf. Artif. Intell. Stat. (AISTATS)* vol. 54. Fort Lauderdale, FL, USA, Apr. 2017, pp. 1273–1282.

[3] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Security*, 2015, pp. 1310–1321.

[4] J. Weng, J. Weng, M. Li, Y. Zhang, and W. Luo, "Deepchain: Auditable and privacy-preserving deep learning with blockchain-based incentive," IACR Cryptol. ePrint Archive, Lyon, France, Rep. 2018/679, 2018.

[5] A. Jochems *et al.*, "Distributed learning: Developing a predictive model based on data from multiple hospitals without data leaving the hospital—A real life proof of concept," *Radiother. Oncol.*, vol. 121, no. 3, pp. 459–467, 2016.

[6] M. Shen, Y. Deng, L. Zhu, X. Du, and N. Guizani, "Privacy-preserving image retrieval for medical IoT systems: A blockchain-based approach," *IEEE Netw.*, vol. 33, no. 5, pp. 27–33, Sep./Oct. 2019.

[7] X. Zhu, H. Li, and Y. Yu, "Blockchain-based privacy preserving deep learning," in *Proc. 14th Int. Conf. Inf. Security Cryptol.*, vol. 11449, 2018, pp. 370–383.

[8] Y. Zhang, Y. Lu, X. Huang, K. Zhang, and S. Maharjan, "Blockchain empowered asynchronous federated learning for secure data sharing in Internet of Vehicles," *IEEE Trans. Veh. Technol.*, vol. 69, no. 4, pp. 4298–4311, Apr. 2020.

[9] Y. Lu, X. Huang, Y. Dai, S. Maharjan, and Y. Zhang, "Blockchain and federated learning for privacy-preserved data sharing in industrial IoT," *IEEE Trans. Ind. Informat.*, vol. 16, no. 6, pp. 4177–4186, Jun. 2020.

[10] B. Hitaj, G. Ateniese, and F. Pérez-Cruz, "Deep models under the GAN: Information leakage from collaborative deep learning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2017, pp. 603–618.

[11] M. Nasr, R. Shokri, and A. Houmansadr, "Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning," in *Proc. IEEE Symp. Security Privacy (SP)* San Francisco, CA, USA, May 2019, pp. 739–753.

[12] L. Melis, C. Song, E. D. Cristofaro, and V. Shmatikov, "Exploiting unintended feature leakage in collaborative learning," in *Proc. IEEE Symp. Security Privacy (SP)*, San Francisco, CA, USA, May 2019, pp. 691–706.

[13] Z. Wang, M. Song, Z. Zhang, Y. Song, Q. Wang, and H. Qi, "Beyond inferring class representatives: User-level privacy leakage from federated learning," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Paris, France, 2019, pp. 2512–2520.

[14] M. Shen, B. Ma, L. Zhu, R. Mijumbi, X. Du, and J. Hu, "Cloud-based approximate constrained shortest distance queries over encrypted graphs with privacy protection," *IEEE Trans. Inf. Forensics Security*, vol. 13, pp. 940–953, 2018.

[15] M. Shen, J. Zhang, L. Zhu, K. Xu, and X. Tang, "Secure SVM training over vertically-partitioned datasets using consortium blockchain for vehicular social networks," *IEEE Trans. Veh. Technol.*, vol. 69, no. 6, pp. 5773–5783, Jun. 2020.

[16] S. Truex *et al.*, "A hybrid approach to privacy-preserving federated learning," in *Proc. 12th ACM Workshop Artif. Intell. Security (AISec CCS)*, 2019, pp. 1–11.

[17] K. Bonawitz *et al.*, "Practical secure aggregation for federated learning on user-held data," 2016. [Online]. Available: arXiv:1611.04482.

[18] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *Proc. Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015.

[19] G. B. Huang, M. Mattar, T. Berg, and E. Learned-Miller, "Labeled faces in the wild: A database forstudying face recognition in unconstrained environments," in *Proc. Dans Workshop Faces Real Life Images Detection Alignment Recognit.*, 2008, pp. 1–14.

[20] S. Li, D. Yi, Z. Lei, and S. Liao, "The CASIA NIR-VIS 2.0 face database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, Portland, OR, USA, 2013, pp. 348–353.

[21] X. Wang, X. Li, S. Pack, Z. Han, and V. C. M. Leung, "STCS: Spatial-temporal collaborative sampling in flow-aware software defined networks," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 6, pp. 999–1013, Jun. 2020.

[22] D. Zhang, X. Chen, D. Wang, and J. Shi, "A survey on collaborative deep learning and privacy-preserving," in *Proc. IEEE 3rd Int. Conf. Data Sci. Cyberspace (DSC)*, Guangzhou, China, 2018, pp. 652–658.

[23] X. Wang, Y. Han, V. C. M. Leung, D. Niyato, X. Yan, and X. Chen, "Convergence of edge computing and deep learning: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 2, pp. 869–904, 2nd Quart., 2020.

[24] M. Shen, J. Duan, L. Zhu, J. Zhang, X. Du, and M. Guizani, "Blockchain-based incentives for secure and collaborative data sharing in multiple clouds," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 6, pp. 1229–1241, Jun. 2020.

[25] R. Li, H. Asaeda, and J. Li, "A distributed publisher-driven secure data sharing scheme for information-centric IoT," *IEEE Internet Things J.*, vol. 4, no. 3, pp. 791–803, Jun. 2017.

[26] X. Wang, Y. Han, C. Wang, Q. Zhao, X. Chen, and M. Chen, "In-edge AI: Intelligentizing mobile edge computing, caching and communication by federated learning," *IEEE Netw.*, vol. 33, no. 5, pp. 156–165, Sep./Oct. 2019.

[27] M. Shen *et al.*, "Blockchain-assisted secure device authentication for cross-domain industrial IoT," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 5, pp. 942–954, May 2020.

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

[29] S. Yeom, I. Giacomelli, M. Fredrikson, and S. Jha, "Privacy risk in machine learning: Analyzing the connection to overfitting," in *Proc. IEEE 31st Comput. Security Found. Symp. (CSF)*, 2018, pp. 268–282.

[30] Y. Long *et al.*, "Understanding membership inferences on well-generalized learning models," 2018. [Online]. Available: arXiv:1802.04889.

[31] J. Hayes, L. Melis, G. Danezis, and E. D. Cristofaro, "LOGAN: Membership inference attacks against generative models," *Proc. Privacy Enhancing Technol.*, vol. 2019, no. 1, pp. 133–152, 2019.

[32] I. Goodfellow *et al.*, "Generative adversarial nets," in *Advances in Neural Information Processing Systems*. Red Hook, NY, USA: Curran, 2014, pp. 2672–2680.

[33] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Security*, 2015, pp. 1322–1333.
[34] G. Ateniese, L. V. 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," *Int. J. Security Netw.*, vol. 10, no. 3, pp. 137–150, 2015.
[35] K. Ganju, Q. Wang, W. Yang, C. A. Gunter, and N. Borisov, "Property inference attacks on fully connected neural networks using permutation invariant representations," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2018, pp. 619–633.
[36] M. Shen, X. Tang, L. Zhu, X. Du, and M. Guizani, "Privacy-preserving support vector machine training over blockchain-based encrypted IoT data in smart cities," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 7702–7712, Oct. 2019.
[37] C. Song, T. Ristenpart, and V. Shmatikov, "Machine learning models that remember too much," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2017, pp. 587–601.
[38] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," 2018. [Online]. Available: arXiv:1807.00459.
[39] F. Schroff, D. Kalenichenko, and J. Philbin, "FaceNet: A unified embedding for face recognition and clustering," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Boston, MA, USA, 2015, pp. 815–823.

Meng Shen (Member, IEEE) received the B.Eng. degree in computer science from Shandong University, Jinan, China, in 2009, and the Ph.D. degree in computer science from Tsinghua University, Beijing, China, in 2014.

He currently serves with Beijing Institute of Technology, Beijing, as an Associate Professor. His research interests include privacy protection for cloud and IoT, blockchain applications, and encrypted traffic classification.

Dr. Shen received the Best Paper Runner-Up Award at IEEE IPCCC 2014.

Huan Wang received the B.Eng. degree in computer science from the Ocean University of China, Qingdao, China, in 2018. She is currently pursuing the master's degree with the Department of Computer Science, Beijing Institute of Technology, Beijing, China.

Her research interests include machine learning and data privacy.

Bin Zhang received the Ph.D. degree from the Department of Computer Science and Technology, Tsinghua University, Beijing, China, in 2012.

He worked as a Postdoctoral Fellow with Nanjing Telecommunication Technology Institute, Nanjing, China, from 2014 to 2017. He is currently a Researcher with the Cyberspace Security Research Center, Peng Cheng Laboratory, Shenzhen, China. He has published more than 40 papers in refereed international conferences and journals. His current research interests focus on network anomaly detection, Internet architecture and its protocols, network traffic measurement, and information privacy security.

Liehuang Zhu (Member, IEEE) received the Ph.D. degree in computer science from Beijing Institute of Technology, Beijing, China, in 2004.

He is a Professor with the Department of Computer Science, Beijing Institute of Technology. He is selected into the Program for New Century Excellent Talents in University from the Ministry of Education, China. His research interests include Internet of Things, cloud computing security, and Internet and mobile security.

Ke Xu (Senior Member, IEEE) received the Ph.D. degree from the Department of Computer Science and Technology, Tsinghua University, Beijing, China, in 2001.

He serves as a Full Professor with Tsinghua University. He has published more than 200 technical papers and holds 11 U.S. patents in the research areas of next-generation Internet, blockchain systems, Internet of Things, and network security.

Prof. Xu has guest-edited several special issues in IEEE and Springer journals. He is an Editor of the IEEE INTERNET OF THINGS JOURNAL. He is the Steering Committee Chair of IEEE/ACM IWQoS. He is a member of ACM.

Qi Li (Senior Member, IEEE) received the Ph.D. degree from Tsinghua University, Beijing, China, in 2012.

He is currently an Associate Professor with the Institute for Network Sciences and Cyberspace, Tsinghua University. He has worked with ETH Zurich, Zürich, Switzerland, and University of Texas at San Antonio, San Antonio, TX, USA. His research interests include network and system security, particularly in Internet and cloud security, mobile security, and big data security.

Dr. Li is currently an Editorial Board Member of the IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING and ACM DTRAP.

Xiaojiang Du (Fellow, IEEE) received the B.S. and M.S. degrees in electrical engineering from Tsinghua University, Beijing, China, in 1996 and 1998, respectively, and the M.S. and Ph.D. degrees in electrical engineering from the University of Maryland at College Park, College Park, MD, USA, in 2002 and 2003, respectively.

He is a tenured Professor with the Department of Computer and Information Sciences, Temple University, Philadelphia, PA, USA. His research interests are wireless communications, wireless networks, security, and systems. He has authored over 400 journal and conference papers in these areas, as well as a book published by Springer.

Dr. Du has been awarded more than $5 million research grants from the U.S. National Science Foundation, Army Research Office, Air Force, NASA, the State of Pennsylvania, and Amazon. He won the Best Paper Award at IEEE GLOBECOM 2014 and the Best Poster Runner-Up Award at ACM MobiHoc 2014. He serves on the editorial boards of three international journals. He is a Life Member of ACM.