NOSnoop: An Effective Collaborative Meta-Learning Scheme Against Property Inference Attack

Xindi Ma, Baopu Li, Qi Jiang, Yimin Chen, Sheng Gao, and Jianfeng Ma, *Member, IEEE*

***Abstract*—Collaborative learning has been used to train a joint model on geographically diverse data through periodically sharing knowledge. Although participants keep the data locally in collaborative learning, the adversary can still launch inference attacks through participants' shared information. In this article, we focus on the property inference attack during model training and design a novel defense mechanism, namely, NOSnoop, to defend such an attack. We propose a collaborative meta-learning architecture to learn the common knowledge over all participants and utilize the natural advantage of meta-learning to hide the sensitive property data. We consider both irrelevant property and relevant property preservation in NOSnoop. For irrelevant property preservation, we utilize the inherent advantage of meta-learning to hide the sensitive property data in meta-training support data set. Thus, the adversary cannot capture the key information related to the sensitive properties and cannot infer victim's private property successfully. For relevant property preservation, an adversarial game is further proposed to reduce the inference success rate of the adversary. We conduct comprehensive experiments to evaluate the effectiveness of NOSnoop. When hiding the sensitive property data in meta-training support data set, NOSnoop achieves an inference AUC score as low as 0.4984 for irrelevant property preservation, meaning the adversary cannot distinguish whether the training batch has the sensitive property data or not. When preserving the relevant property, NOSnoop is able to achieve an inference AUC score of 0.5091 without compromising model utility.**

***Index Terms*—Inference attack, machine learning, meta-learning, privacy preservation.**

Manuscript received January 18, 2021; revised June 17, 2021; accepted September 10, 2021. Date of publication September 15, 2021; date of current version April 25, 2022. This work was supported in part by the National Natural Science Foundation of China under Grant 61902290, Grant 61902291, Grant 62072487, Grant 62072352, and Grant 61872283; in part by the Key Research and Development Program of Shaanxi under Grant 2020ZDLGY09-06 and Grant 2019ZDLGY12-04; in part by the National Statistical Science Foundation of China under Grant 2020LD01; in part by the Scientific Research Program Funded by Shaanxi Provincial Education Department under Grant 20JY016; and in part by the Beijing Natural Science Foundation under Grant M21036. *(Corresponding author: Sheng Gao.)*

Xindi Ma and Jianfeng Ma are with the School of Cyber Engineering, Xidian University, Xi'an 710071, China (e-mail: xdma@xidian.edu.cn; jfma@mail.xidian.edu.cn).

Baopu Li is with Baidu Research, Sunnyvale, CA 94089 USA (e-mail: baopuli@baidu.com).

Qi Jiang is with the School of Cyber Engineering, Xidian University, Xi'an 710071, China, and also with the Network Communication Research Centre, Peng Cheng Laboratory, Shenzhen 518055, China (e-mail: jiangqixdu@gmail.com).

Yimin Chen is with the Department of Computer Science, Virginia Polytechnic Institute and State University, Northern Virginia Center, Falls Church, VA 22043 USA (e-mail: yiminchen@vt.edu).

Sheng Gao is with the School of Information, Central University of Finance and Economics, Beijing 100081, China (e-mail: sgao@cufe.edu.cn).

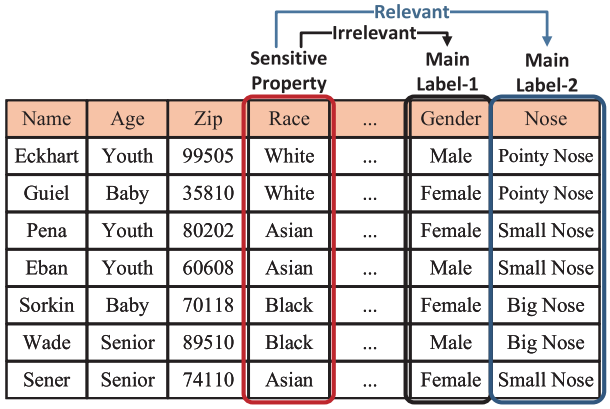
Digital Object Identifier 10.1109/JIOT.2021.3112737

I. Introduction

With the development of Internet of Things (IoT), the smart IoT devices have been integrated into all aspects of our lives (e.g., smart cameras). In order to make better use of collected data and provide more intelligent decision making, AI technology has been applied to IoT applications to achieve the real-time edge intelligence. However, the increasing awareness of data privacy (e.g., faces, identities, and behavioral habits) motivates users to keep data locally on their own devices. Privacy concern on large-scale data aggregation also leads to administrative policies, such as general data protection regulation (GDPR) in European Union and California Privacy Act (CPA) in USA. Together with the greatly enhanced computation capability of end devices, collaborative learning (termed as federated learning as well) emerges as a vastly developed learning scheme in real-world IoT applications. Collaborative learning only requires gradients rather than raw data from participants for model training; hence, data privacy protection comes naturally with little cost. Unfortunately, researchers [1]–[3] found that an attacker could still infer private information of participants in collaborative learning merely from shared knowledge, such as gradients, empirical loss, or model parameters.

As a typical inference attack during training process, the property (or attribute) inference attack is to infer whether a training batch has data with a specific property (which is usually sensitive, e.g., race) or not [3]. To tackle this property inference attack, many schemes based on cryptographic tools or differential privacy have been proposed. However, most of them are not yet for practical usage. Cryptographic schemes usually need to outsource the calculation to two or more non-colluding servers, which may be unrealistic in practice or have privacy issues while using only a single server [4], [5]. Although differential privacy is able to protect participants' data privacy, it comes with the cost of model utility [6].

When considering privacy protection in collaborative learning, a key question is *which part of knowledge should be learned and transferred to other participants*. Here, we consider the scenarios where participants' data sets are heterogeneous. Hence, participants prefer to learn personalized models for different individuals than to learn an identical model for all. Also, they tend to prioritize protection of private properties. Thus, we focus on collaborative meta-learning architecture, which is often referred as federated meta-learning [7], [8]



Name	Age	Zip	Race	...	Gender	Nose
Eckhart	Youth	99505	White	...	Male	Pointy Nose
Guiel	Baby	35810	White	...	Female	Pointy Nose
Pena	Youth	80202	Asian	...	Female	Small Nose
Eban	Youth	60608	Asian	...	Male	Small Nose
Sorkin	Baby	70118	Black	...	Female	Big Nose
Wade	Senior	89510	Black	...	Male	Big Nose
Sener	Senior	74110	Asian	...	Female	Small Nose

Fig. 1. Samples of training data. *Race* is the sensitive property to protect, and *gender* and *nose* are two main labels for two different prediction models. For prediction of *gender*, *race* is irrelevant with the main label *gender*. For prediction of *nose*, *race* is relevant with the main label *nose*.

and is to train a common initialization model for all participants, where the common initialization model is generally referred as meta-model. In collaborative meta-learning, each participant trains its model locally and exchanges its trained knowledge periodically to update the meta-model. Once the training of meta-model is complete, each participant can quickly obtain her/his own personalized model by retraining the meta-model on her/his own samples. One key point to note is that meta-learning brings in the unique benefit of fast adaptation, greatly reducing the number of required samples for retraining. Compared with the traditional machine learning, meta-learning includes inner-loop training and outer-loop training and the inner-loop training of each participant is invisible to the other ones. Thus, meta-learning has the natural advantage in removing the sensitive-property data from the outer-loop training data. Although the adversaries eavesdrop the sharing knowledge, it is difficult for them to infer the sensitive property information successfully.

The property inference attack does not fade away in collaborative meta-learning. The challenge to defend against such attacks roots in the strong adversary model that the attacker can obtain internal knowledge, e.g., gradients, of the training process. In this article, we propose our novel privacy protection mechanism, NOSnoop, to tackle such powerful attack. We assume that each sample has a main prediction label and a sensitive property to protect, as shown in Fig. 1. Noting that the sensitive property can be relevant or irrelevant with the corresponding main label, we design NOSnoop considering the following two cases. *Case I:* For irrelevant property protection, the sensitive property does not contribute to the training of the main label, as illustrated in Fig. 2(a). This enables us to purposely restrain samples with sensitive property in meta-training support data set only rather than in query data set as well. In this way, the best that an attacker can do is to exploit gradients from meta-training query data set for inference, resulting in a much lower success rate. *Case II:* For relevant property protection, samples with sensitive property need to occur in both meta-training support and query data set to boost the prediction performance of main label, as shown in Fig. 2(b). For example, the *race* property may help to predict people's *nose size*. In this case, we propose to use an

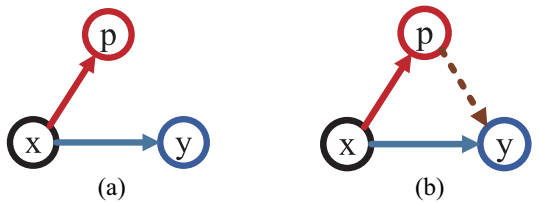


Fig. 2. Illustration for (a) irrelevant and (b) relevant properties. x is the observation data, y is the main label to predict, and p is the sensitive property that victims want to preserve.

adversarial game to reduce the impact of sensitive property as much as possible from the shared knowledge, thus reducing the success rate of the attacker. In heterogeneous data, we can still split the property-preservation data into relevant and irrelevant sensitive property data. While training the model, all participants have the same targets to train a collective model. Hence, the main label is the same for all participants. However, these participants' sensitive properties may be different. Therefore, different participants can independently choose corresponding privacy preserving mechanisms according to the relationship between their sensitive properties and the main label. The main contributions of this article are as follows.

1) *Collaborative Meta-Learning Against Property Inference Attack:* We propose the collaborative meta-learning to learn the common knowledge over all participants and hide their sensitive property data. So far as we know, this is the first work that utilizes meta-learning to defend against property inference attack in collaborative learning.
2) *Sensitive Property Preservation:* For irrelevant property protection, we advocate that only the meta-training support data set is related to the sensitive property data. Thus, the shared knowledge is only calculated on the nonsensitive property data (meta-training query data set), making it difficult for the adversary to infer the sensitive property. For relevant property protection, we introduce a feature extractor to obtain the related features and design a discriminator to detect the sensitive property. Then, an adversarial game is proposed between feature extractor and discriminator to reduce the sensitive property representation from the shared knowledge.
3) *Experimental Validation:* We comprehensively evaluate NOSnoop on two real-world data sets. For the irrelevant property preservation, the property inference attack failed with an inference AUC score of 0.4984. For the relevant property preservation, we obtain the inference AUC score of 0.5091 with model accuracy of 0.8. We also evaluate the nonprivacy-preserving (NPP) mechanism and differentially private (DP) mechanism, resulting in 0.9296 AUC score with 0.8264 model accuracy and 0.5232 AUC score with 0.5285 model accuracy $((5, 10^{-5})$-DP), respectively. Our experimental results confirm the effectiveness of NOSnoop and its superiority over NPP and DP.

II. Related Work

In recent years, privacy leakage has gained widespread concern in both academia and industry, such as the location

privacy [9]–[11], social privacy [12], [13], identity privacy [14], [15], and so on. With the development of machine learning, the privacy issues in machine learning have also received a lot of attention and many valuable works have been presented [16]–[18].

A. Privacy-Preserving Machine Learning

Machine learning includes centralized learning and distributed learning. In centralized learning, the service provider (model trainer) collects the training data from data owners and trains the model on the joint data set. In distributed learning, data owners take part in the model training, and a global model is learned collaboratively through sharing knowledge. Although the distributed learning does not need the participants to share their training data, it still faces the privacy issues during the model training [19]–[22]. Wei *et al.* [19] presented a principled framework for evaluating and comparing different forms of client privacy leakage attacks. They provided formal and experimental analysis to show how adversaries could reconstruct the private local training data by simply analyzing the shared parameter update from local training. Thus, many privacy-preserving mechanisms have been designed to defend such attacks in centralized or distributed learning. To protect the privacy in centralized learning, Wang *et al.* [23], Mohassel and Zhang [24], and Wagh *et al.* [25] proposed several protocols based on the secure multiparty computations to achieve the model training over encrypted real numbers. However, they did not consider the collected data that typically comes from several data owners and is encrypted with different keys. To achieve the model training on multikey encrypted data, Li *et al.* [26] and Ma *et al.* [4] introduced two privacy-preserving mechanisms based on the homomorphic encryption. Recently, distributed learning, such as collaborative learning and federated learning [27], has become popular and many privacy-preserving distributed learning architectures have been designed [1], [2], [28]. Abadi *et al.* [2] advanced a DP technique to add Gaussian noise to the shared gradients and proposed moments accountant technique to analyze the privacy loss. Xie *et al.* [29] and Zhou *et al.* [30] also came up with two DP mechanisms to tackle the data attacks. Although these DP mechanisms protected the privacy well, Truex *et al.* [6] showed that the tradeoff between classification accuracy and attack vulnerability could not be achieved at the same time. Phong *et al.* [5] put forward a privacy-preserving aggregation algorithm for parameter gradients based on additively homomorphic encryption. However, participants' local data can still be surreptitiously extracted from two adjacent versions of parameters.

Apart from schemes based on differential privacy and cryptography, there are other methods to protect privacy in machine learning. Malekzadeh *et al.* [31] introduced a replacement autoencoder to protect user's privacy. The autoencoder learned how to transform discriminative features of data that corresponded to sensitive instances into some features that had been more observed in nonsensitive instances. Nasr *et al.* [32] constructed a Stackelberg game to protect the membership privacy. Yang *et al.* [33] also built a purification framework to defend the membership inference attack. However, these systems cannot handle property inference attacks well. We compare these existing schemes with NOSnoop in Table I.

TABLE I
COMPARISON SUMMARY

Function / Method	[4]	[2]	[32]	[5]	NOSnoop
Collaborative learning	×	✓	×	✓	✓
Privacy-preserving model training	✓	✓	✓	×	✓
Property privacy	✓	✓	N.A.	×	✓
Model utility	High	Low	Low	High	High
Training efficiency	Low	High	High	High	High

B. Inference Attack in Machine Learning

There are two main types of inference attacks in machine learning: 1) membership inference [34], [35] and 2) property inference [3], [36]. Shokri *et al.* [35] demonstrated the membership inference attack by designing several shadow models to learn the difference of target model's outputs influenced by different input samples. Yu *et al.* [37] considered the membership inference as a double-edged sword and utilized some private augmented data to sharpen its good side while inhibiting its bad side. Hidano *et al.* [38] proposed a transfer shadow training technique, where an adversary employed the parameters of the transferred model to construct shadow models, to significantly improve the performance of membership inference when a limited amount of shadow training data was available to the adversary. Chen *et al.* [20] proposed the first generic membership inference attack model that could be instantiated in a large range of settings and was applicable to various kinds of deep generative models.

For the property inference, one scenario is to infer whether a given sample has the hidden sensitive property with black-box access to the trained model [36]. Another is to infer whether a training batch has the sensitive property data during the model training [3]. Melis *et al.* [3] designed passive and active property inference attacks by analyzing the collected gradients during model training. Luo *et al.* [39] also presented several property inference attack methods to investigate the potential privacy leakages in the model prediction stage of vertical federated learning. In this article, we focus on the second property inference scenario and propose a novel mechanism to defend the attack.

C. Collaborative Meta-Learning

Collaborative meta-learning is also referred as federated meta-learning. The concept of meta-learning has been proposed for many years. But the advances of gradient-based optimization bring it into the light again as a promising solution for fast learning. In particular, Finn *et al.* [40] proposed one gradient-based algorithm, called model agnostic meta-learning (MAML), which directly optimizes the learning performance with respect to an initialization of the model such that even one-step gradient decent from that initialization can still produce good results on a new task. Similar to the federated learning, meta-learning is also a multitask learning. However, federated learning tries to train a global

model that fits the data as accurately as possible and is not designed to achieve fast learning over small data sets. Thus, federated meta-learning is proposed and tries to achieve fast adaptation and good performance over even small training data set. Chen *et al.* [41] adopted meta-learning to handle some bottlenecks of federated learning in real-world applications. They designed a federated meta-learning framework to limit the scale of shared knowledge, so as to achieve efficient communication and rapid convergence. Lin *et al.* [8] proposed a platform-aided collaborative meta-learning framework to achieve real-time edge intelligence in IoT applications. They investigated the convergence of the proposed collaborative meta-learning algorithm under mild conditions on node similarity and the adaptation performance at the target edge. Fan and Huang [42] also proposed a federated few-shot learning framework to learn a few-shot classification model that can classify unseen data classes with only a few labeled samples. With the federated learning strategy, the designed system could utilize many data sources while keeping data privacy and communication efficiency.

III. Preliminaries

A. Model Agnostic Meta-Learning

As described above, the key question of privacy-preserving collaborative learning is what knowledge should be learned and transferred. Since meta-learning has a strong generalization ability, we propose to use it to learn the common knowledge over all participants' data sets and share this learned knowledge to achieve collaborative learning. Meta-learning includes two phases: 1) meta-training and 2) meta-adaption [43]. Meta-training is used to learn the common knowledge from all tasks. Based on the learned knowledge, meta-adaption can quickly adapt to a new task using only a few samples.

While training the model using meta-learning, the training data set consists of two parts: 1) meta-training part, denoted by $\mathcal{D}_{\text{train}}$ and 2) meta-adaption part $\mathcal{D}_{\text{adapt}}$. Each meta-training task or meta-adaption task also has two data sets: 1) support data set and 2) query data set. For meta-training tasks, we denote their support data set and query data set as $\mathcal{D}^s_{\text{train}}$ and $\mathcal{D}^q_{\text{train}}$, respectively. Similarly, the support data set and query data set for meta-adaption tasks are $\mathcal{D}^s_{\text{adapt}}$ and $\mathcal{D}^q_{\text{adapt}}$. As shown in Fig. 3, the meta-training process includes inner-loop update (base-model update) and outer-loop update (meta-model update). Let us take the ith meta-training task T_i as an example. T_i's support and query data set are $\{\mathcal{D}^s_{\text{train},i}, \mathcal{D}^q_{\text{train},i}\}$. At the beginning, T_i initializes the base model with the current meta-model Θ. Then, T_i performs the inner-loop update as follows and obtains a trained base-model θ_i based on the support data set $\mathcal{D}^s_{\text{train},i}$:

$$\theta_i \leftarrow \theta_i - \eta_1 \nabla_{\theta_i} \frac{1}{\left|\mathcal{D}^s_{\text{train},i}\right|} \sum_{\{x,y\}\in\mathcal{D}^s_{\text{train},i}} L_i\big(f_{\theta_i}(x), y\big) \tag{1}$$

where $f_{\theta_i}(\cdot)$ is the base model, $L_i(\cdot)$ is the loss function for T_i in inner loop, and η_1 is the learning rate in inner loop update.

After that, T_i evaluates the trained base model θ_i on the query data set $\mathcal{D}^q_{\text{train},i}$ and obtains the empirical loss

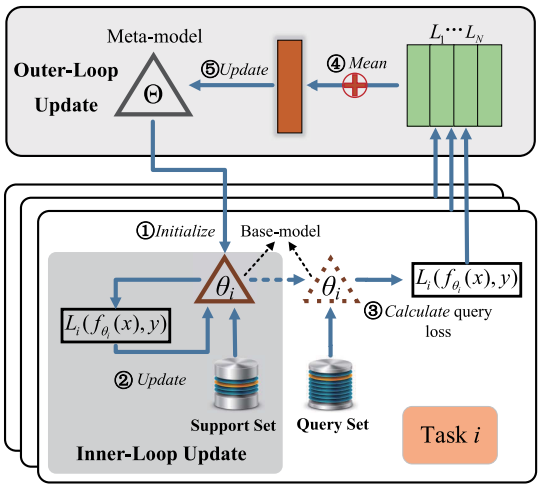


Fig. 3. Overview of the meta-training process.

TABLE II
Definitions and Notations in NOSnoop

Symbol	Definition
$\mathcal{D}^s_{\text{train},i}, \mathcal{D}^q_{\text{train},i}$	T_i's support and query dataset in meta-training tasks
$\mathcal{D}^s_{\text{adapt},i}, \mathcal{D}^q_{\text{adapt},i}$	T_i's support and query dataset in meta-adaption tasks
N	Number of meta-training tasks (participants in NOSnoop)
θ_i	T_i's base-model parameters
Θ	Parameters in meta-model
θ_{fea}	Parameters in feature extractor
θ_{pre}	Parameters in main-label predictor
θ_{disc}	Parameters in sensitive-property discriminator

$L_i(f_{\theta_i}(x), y)$. $L_i(f_{\theta_i}(x), y)$ measures the quality of base model θ_i in terms of the total loss of using meta-model Θ across all tasks. This meta objective is now minimized to optimize the model parameters Θ. It is these parameters Θ that contains the across-task knowledge. The optimization of this meta objective is called the outer-loop update process. Next, meta-learning aggregates all the query losses calculated in the meta-training tasks and updates the meta-model Θ in the outer loop

$$\Theta \leftarrow \Theta - \eta_2 \nabla_{\Theta} \frac{1}{N} \sum_{i=1}^{N} \frac{1}{\left|\mathcal{D}^q_{\text{train},i}\right|} \sum_{\{x,y\}\in\mathcal{D}^q_{\text{train},i}} L_i\big(f_{\theta_i}(x), y\big) \tag{2}$$

where η_2 is the learning rate for outer loop update, N is the number of meta-training tasks, $f_{\theta_i}(\cdot)$ is the trained base model in inner loop, and $L_i(f_{\theta_i}(x), y)$ is the loss of $f_{\theta_i}(\cdot)$ on T_i's query data set $\mathcal{D}^q_{\text{train},i}$.

When adapting the meta-model to new tasks, we can also use $\mathcal{D}^s_{\text{adapt},i}$ (usually a small data set) to update the meta-model Θ and evaluate the trained model on $\mathcal{D}^q_{\text{adapt},i}$. Finally, we can obtain a personalized model relying on the meta-adaption data set. For more clarity, some notations are listed in Table II.

B. Property Inference Attack

In collaborative learning, an adversary can launch inference attacks on participant's shared knowledge and obtain sensitive information [3], [5]. In this article, we consider the property inference attack during model training. We assume that the

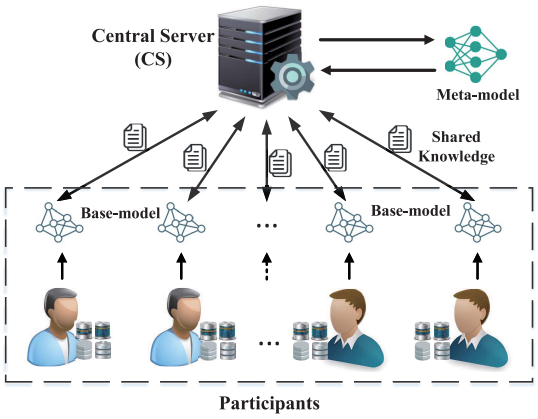


Fig. 4. Overview of NOSnoop.

adversary has *auxiliary data* consisting of sensitive property data points and nonsensitive property ones, the same as [3].

Property inference attacks can be passive or active attacks. In the passive property inference attack, the adversary can join the model training and generate aggregated updates over the auxiliary data points. The attack is passive as the adversary just observes the updates and does not change the model training process including the local and global collaborative training.

In the active property inference attack, the adversary can modify its local model architecture to launch an attack. We assume that each record has a main label y for main task and a sensitive property label c. The adversary integrates its local model with an augmented property classifier and joins the model loss as follows [3]:

$$L_{\text{adv}} = \alpha L\big(f_{\theta_{\text{adv}}}(x), y\big) + (1-\alpha)L\big(f_{\theta_{\text{adv}}}(x), c\big) \tag{3}$$

where α is a normalization parameter, $f_{\theta_{\text{adv}}}(\cdot)$ is the current trained model of the adversary, $L(\cdot)$ is the loss function for collaborative training, and $\{x; y, c\}$ is the training data set with main label y and property label c. As indicated in [3], the above L_{adv} caused the trained model to learn separable representations for data records with and without the sensitive property.

IV. System Overview and Problem Formulation

A. Framework of NOSnoop

As shown in Fig. 4, our proposed NOSnoop system comprises two parts: 1) central server (CS) and 2) participants. To achieve the privacy preservation in collaborative learning, we introduce collaborative meta-learning to learn the across-task knowledge and obtain a generalization meta-model. Then, participants can further train personalized models over their meta-adaption data sets.

1) *CS* manages the global parameters of the collaborative training model (i.e., the meta-model in meta-learning) and performs the outer loop update training based on the received empirical loss from participants. After completing the meta-training, CS releases the meta-model parameters to participants to perform meta-adaption.
2) *Participants* own their local training data (including meta-training data and meta-adaption data) and they have the same purpose for collaborative learning. At the beginning of meta-training, participants initialize the base model with the downloaded parameters from CS. Then, participants perform the inner loop update over their meta-training support data set and calculate the empirical loss over the meta-training query data set. After that, participants upload the query loss to CS to update the meta-model. Finally, after the meta-training, participants download the meta-model from CS and retrain their personalized model on their meta-adaption data sets.

In NOSnoop, the secure socket layer (SSL) or transport layer security (TLS) protocol is adopted to secure all communications, which ensures the data integrity and authenticity between two communication entities.

B. Adversary Model

Here, we assume that CS and participants are *honest-but-curious*. Although they strictly perform the model training, they are also interested in gathering or learning other parties' private information during the training process. Based on such an assumption, we introduce an active adversary $\mathcal{A}$ that has the *auxiliary data* sampled from the same class as the victim's data and launches property inference attack to infer whether the sensitive property data appear in the training batch or not. The adversary $\mathcal{A}$ has the following capabilities.

1) $\mathcal{A}$ can *eavesdrop* all communications to obtain the shared knowledge and launch an *active attack* to intercept and forge the transmitted messages.
2) $\mathcal{A}$ can *compromise* CS to observe the outer loop update process for meta-model. It can collect the received losses from participants and launch the passive property inference attack.
3) $\mathcal{A}$ can *compromise* one or more participants, except the victims, to launch active or passive property inference attack as described in Section III-B. However, $\mathcal{A}$ is still honest that $\mathcal{A}$ follows the collaborative model training protocol and submits the nonmalformed messages.

C. Privacy Requirement

In NOSnoop, we try to protect participants' sensitive property in training data. Although the membership inference is a typical inference attack, its goal is to infer whether a given record appears in the training data set or not and reveal whether this record has the sensitive property. This membership inference attack has been demonstrated that the attack accuracy is often misleading and a simple blind attack, which is highly unreliable and inefficient in reality, can often represent similar accuracy [44]. Thus, in this article, we focus on the property inference attack during collaborative model training. Our NOSnoop should preserve participants' sensitive property and make the adversary in Section IV-B fail to distinguish whether a training batch has sensitive property data records or not.

V. Details of NOSnoop

In this section, we present the details of NOSnoop. Since the across-task knowledge can be learned in meta-learning and

can quickly adapt to the new tasks using only a few samples, we believe that meta-learning has strong generalization ability for the meta-training tasks. Hence, in collaborative learning, we treat each participant as a task node and participants perform the meta-training tasks on their local training data. Thus, the shared common knowledge can be learned through meta-training and the collaborative model (meta-model in meta-learning) is trained after several epochs. Afterward, each participant trains its personalized model using meta-adaption on the meta-adaption data set. Since the information sharing just occurs in meta-training, the adversary can only collect the shared loss in meta-training and observe nothing in meta-adaption. In the following, we will first discuss how to protect the sensitive irrelevant and relevant properties and then introduce the passive and active property inference attack in NOSnoop. Since all participants can adopt the same strategy to achieve the sensitive property preservation, we take participant T_i as an example to perform the privacy preservation in the following sections.

Algorithm 1: Irrelevant Property Preservation

Data: Training datasets $\mathcal{D}_i$ and sensitive property p.
Result: Participant T_i's personalized model.
($@T_i$): split training dataset into $\mathcal{D}_{\text{train},i}$ and $\mathcal{D}_{\text{adapt},i}$;
($@T_i$): filter out $\mathcal{D}_{\text{sens},i}$ from $\mathcal{D}_{\text{train},i}$ and the remaining set denoted by $\mathcal{D}_{\text{nonsens},i}$;
($@T_i$): split $\mathcal{D}_{\text{nonsens},i}$ into two parts: $\mathcal{D}^s_{\text{nonsens},i}$ and $\mathcal{D}^q_{\text{nonsens},i}$;
($@T_i$): construct $\mathcal{D}^s_{\text{train},i} = \mathcal{D}_{\text{sens},i} \cup \mathcal{D}^s_{\text{nonsens},i}$ and $\mathcal{D}^q_{\text{train},i} = \mathcal{D}^q_{\text{nonsens},i}$;
while *True* **do**
    ($@T_i$): initialize the base-model using the latest meta-model, train the base-model over $\mathcal{D}^s_{\text{train},i}$ in inner-loop;
    ($@T_i$): evaluate the trained base-model on $\mathcal{D}^q_{\text{train},i}$;
    (@CS): aggregate the query loss received from all participants and update the meta-model;
end
($@T_i$): retrain the base-model on $\mathcal{D}_{\text{adapt},i}$ to obtain the personalized model.

A. Irrelevant Property Preservation

As illustrated in Fig. 2(a), since sensitive property p is irrelevant with main task label y, the extracted features related to property p have little influence on the training accuracy of the main task. For example, when we train a model to predict the *gender* (the main label) of a person, the sensitive property *race* which victims want to preserve is irrelevant with the label *gender*. Hence, we can hide the training data that has sensitive property p to reduce the success rate of the property inference attack.

In meta-training, T_i has the support data set $\mathcal{D}^s_{\text{train},i}$ and query data set $\mathcal{D}^q_{\text{train},i}$ to train the base model. T_i performs the inner loop update over $\mathcal{D}^s_{\text{train},i}$ and calculates the query loss over $\mathcal{D}^q_{\text{train},i}$. So if the adversary collects the shared knowledge, it just observes the empirical loss, which is directly related to T_i's query data set $\mathcal{D}^q_{\text{train},i}$ and indirectly related to the support data set $\mathcal{D}^s_{\text{train},i}$. Therefore, we can use the support data set to conceal the training data that has the sensitive property.

Before starting the meta-training, T_i filters out the data records that have the sensitive property from the meta-training data set, denoted by $\mathcal{D}_{\text{sens},i}$. Then, the remaining data set that does not have the sensitive property is denoted by $\mathcal{D}_{\text{nonsens},i}$. To construct the support data set and query data set, T_i also splits the nonsensitive property data set $\mathcal{D}_{\text{nonsens},i}$ into two parts: 1) $\mathcal{D}^s_{\text{nonsens},i}$ and 2) $\mathcal{D}^q_{\text{nonsens},i}$. Thus, the support data set for meta-training is $\mathcal{D}^s_{\text{train},i} = \mathcal{D}_{\text{sens},i} \cup \mathcal{D}^s_{\text{nonsens},i}$ and the query data set is $\mathcal{D}^q_{\text{train},i} = \mathcal{D}^q_{\text{nonsens},i}$. As such, in the inner loop update, T_i randomly chooses a training batch from the support data set $\mathcal{D}^s_{\text{train},i}$, and may have or may not have sensitive property data, to feed the base model. After several iterations, T_i randomly chooses a data batch from the query data set $\mathcal{D}^q_{\text{train},i}$ to evaluate the trained base model and uploads the calculated loss to CS. Since the query data set does not have any sensitive property data record, the uploaded loss is only directly related to the nonsensitive property data and it is difficult for the adversary to infer whether this training epoch used the sensitive property data or not. We summarize the irrelevant property preservation in Algorithm 1.

B. Relevant Property Preservation

When training the model of which the main prediction label is relevant to the sensitive property as illustrated in Fig. 2(b), we may reveal the sensitive property data to learn the across-task knowledge to increase the performance. If we still conceal the data labeled with sensitive property in support data set (as discussed in Section V-A), the trained model may have a poor utility because of missing the knowledge about sensitive property p. Thus, while training the main task, we should evenly distribute the sensitive property data into the support data set and query data set for the meta-training. In this case, this constructed query data set may greatly increase the privacy leakage of the sensitive property. To defend against the property inference attack in relevant property model training, we construct an adversarial game and design a discriminator to help to conceal the sensitive property p.

We first preprocess the training samples in meta-training data set $\mathcal{D}_{\text{train},i}$ and add an additional label c of the sensitive property whether the sample has ($c = 1$) or does not have ($c = 0$). Then, a feature extractor F is constructed to extract the features from the training samples. Next, a main-label predictor P and a sensitive property discriminator D are also designed to achieve the main label prediction and reduce the sensitive property representation in shared knowledge. We, respectively, present the designs of F, P, and D as follows.

1) Feature Extractor F: In NOSnoop, the feature extractor is used to extract the feature representations for main label and sensitive property over the representation space. Thus, when we input the meta-training data $\{x, y\} \in \mathcal{D}_{\text{train},i}$, we want to extract the representation $m = F_{\theta_{\text{fea},i}}(x)$ to preserve the features, which are necessary to predict the main label

y while reducing the features related to the sensitive property p. When evaluating NOSnoop with neural network in Section VII, we utilize the first two convolutional layers of constructed VGGNet as the feature extractor. Since the convolutional layer can learn the generic image features [45], we use the designed feature extractor to learn the representation related to main label y and sensitive property p simultaneously. Then, based on the extracted features m, we can train P to predict main label y, denoted by $P_{\theta_{\text{pre},i}}(m)$, and train D to predict sensitive property label c, denoted by $D_{\theta_{\text{disc},i}}(m)$.

2) Main-Label Predictor P: The predictor is designed to achieve the main label prediction on meta-training data set. To obtain a high performance to predict the main label, we need to minimize the empirical loss to optimize the predictor and feature extractor. Thus, we define T_i's main-label prediction loss as

$$L_{\text{pre},i} \leftarrow \sum_{\{x,y\}\in\mathcal{D}_{\text{train},i}} L_i\big(P_{\theta_{\text{pre},i}}\big(F_{\theta_{\text{fea},i}}(x)\big), y\big). \tag{4}$$

3) Sensitive-Property Discriminator D: The discriminator is trained to predict the sensitive property label c based on the extracted features m. However, to tackle the property inference attack, discriminator D needs to protect the private information related to p while achieving a good performance to predict the sensitive property label c. It means that we need to reserve the invariance and eliminate the variations of label c from the extracted features m. Thus, we introduce an adversarial process [46] to enable the feature extractor F to minimize the loss of the sensitive property prediction. Simultaneously, the sensitive property discriminator D also fights to maximize the same loss of predicting the sensitive property label. Intuitively, the feature extractor and the discriminator form an adversarial game where the feature extractor tries to conceal the label c but the discriminator tries to detect it. In the evaluation, we utilize the latter two convolutional layers of VGGNet as the discriminator. The discriminator loss is defined as follows:

$$L_{\text{disc},i} \leftarrow \sum_{\{x,c\}\in\mathcal{D}_{\text{train},i}} L_i\big(D_{\theta_{\text{disc},i}}\big(F_{\theta_{\text{fea},i}}(x)\big), c\big). \tag{5}$$

Formally, F, P, and D jointly play the following adversarial game:

$$\min_{F,P}\max_{D} \mathcal{L}(F,P,D) \tag{6}$$

where $\mathcal{L}(F,P,D) = L_{\text{pre},i} - \lambda L_{\text{disc},i}$, λ is a hyperparameter to adjust the strength of the discrimination.

To achieve the adversarial game, we introduce a gradient reversal layer, which was proposed by Ganin and Lempitsky [46] to train all the three components together with a standard machine learning optimizer. Mathematically, gradient reversal layer can be treated as a "pseudofunction" $R_\lambda(x)$, which has the following properties:

$$\begin{aligned} R_\lambda(x) &= x \\ \frac{dR_\lambda}{dx} &= -\lambda\mathbf{I} \end{aligned} \tag{7}$$

where $\mathbf{I}$ is an identity matrix. Then, we reformulate the empirical loss $\mathcal{L}(F,P,D)$ as follows:

$$\begin{aligned} \mathcal{L}(F,P,D) \leftarrow &\sum_{\{x,y\}\in\mathcal{D}^s_{\text{train},i}} L_i\big(P_{\theta_{\text{pre},i}}\big(F_{\theta_{\text{fea},i}}(x)\big), y\big) \\ &+ \sum_{\{x,c\}\in\mathcal{D}^s_{\text{train},i}} L_i\big(D_{\theta_{\text{disc},i}}\big(R_\lambda\big(F_{\theta_{\text{fea},i}}(x)\big)\big), c\big). \end{aligned} \tag{8}$$

Then, we can use the standard optimization algorithm to train our model. We summarize the relevant property preservation in Algorithm 2.

Algorithm 2: Relevant Property Preservation

Data: Training datasets $\mathcal{D}_i$ and sensitive property p.
Result: Participant T_i's personalized model.

($@T_i$): split training dataset into $\mathcal{D}_{\text{train},i}$ and $\mathcal{D}_{\text{adapt},i}$;
($@T_i$): add an additional label c for each sample in $\mathcal{D}_{\text{train},i}$ to indicate its sensitive property label;
($@T_i$): randomly spilt $\mathcal{D}_{\text{train},i}$ into $\mathcal{D}^s_{\text{train},i}$ and $\mathcal{D}^q_{\text{train},i}$;
($@T_i$): construct the feature extractor F, main-label predictor P, and sensitive-property discriminator D;
($@T_i$): add a gradient reversal layer between F and D;
while *True* **do**
  ($@T_i$): update the base-model with adversarial game on $\mathcal{D}^s_{\text{train},i}$:
  $$\theta_{\text{fea}}, \theta_{\text{pre}}, \theta_{\text{disc}} \leftarrow \min_{F,P}\max_{D} \mathcal{L}(F,P,D);$$
  ($@T_i$): calculate the empirical loss on $\mathcal{D}^q_{\text{train},i}$;
  (@CS): aggregate the query loss received from all participants and update the meta-model;
end
($@T_i$): retrain the base-model on $\mathcal{D}_{\text{adapt},i}$ to obtain the personalized model.

C. Passive and Active Property Inference Attack

In our adversary model, we assume the adversary could compromise CS or participants. If the adversary compromises CS, it could collect victim's empirical loss calculated on the meta-training query data set and can only launch the passive property inference attack. While compromising participants, the adversary could recover victim's empirical loss or the aggregated loss over several participants and launch the passive or active inference attack. If the system just has two participants, one of them is the adversary and the other one is the victim. In this case, the adversary can recover the victim's loss calculated on its meta-training query data set. But if there are three or more honest participants in the system, the adversary will only recover the aggregated loss.

To judge whether the training batch has the sensitive property data or not, the adversary needs to construct a *binary property classifier*, denoted by f_{sens}. Then, before starting the inner-loop update, the adversary first snapshots the base model (or meta-model) at each epoch, denoted by θ_s, and prepares the training data for f_{sens} based on its auxiliary data mentioned in Sections III-B and IV-B. With the snapshot model θ_s, the adversary chooses a data batch with the sensitive property

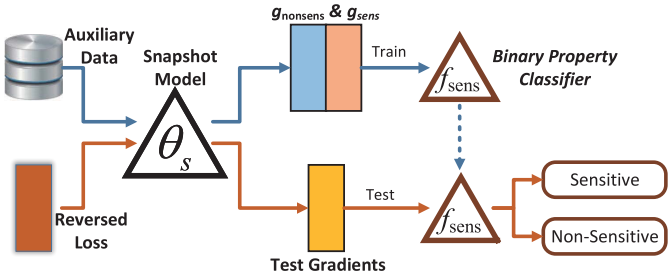


Fig. 5. Pipeline of property inference attack.

from its auxiliary data and calculates the parameter gradients, denoted by g_{sens}. Then, a nonsensitive property data batch is also chosen to calculate the gradients, denoted by g_{nonsens}. The adversary labels 1 and 0 for g_{sens} and g_{nonsens}, respectively, and collects these gradients as training data for f_{sens}. Next, based on the reversed loss, the adversary also calculates the parameter gradients over the snapshot model θ_s and stores the gradients as the test data for f_{sens}. After the meta-training, the adversary trains f_{sens} on the collected training gradients and obtains the attack results by testing the trained f_{sens} on the collected testing gradients. For clarification, we illustrate the property inference attack pipeline in Fig. 5.

VI. Scalability of NOSnoop

Although NOSnoop can resist the property inference attack launched through the shared knowledge, many other inference attacks may cannot be defended, such as the membership inference attack, data recovery attack, model inversion attack, and so on. Hence, we can adopt a more stringent privacy-preserving mechanism to protect the model training and release process. As two typical privacy-preserving mechanisms, differential privacy and homomorphic encryption are usually designed into the training process.

To protect the model training process, we can use the homomorphic encryption (or differential privacy) to encrypt (or sanitize) the shared empirical loss. If we sanitize the shared loss by differential privacy, CS can perform the outer loop model update as usual. Although the added noise may reduce the model accuracy, this sanitized method can resist many inference attack, such as the membership inference attack, model inversion attack, and so on. While using the homomorphic encryption to protect the empirical loss, each participant needs to encrypt the shared knowledge before uploading and CS performs the outer loop update over encrypted loss and parameter gradients. To achieve the model update over ciphertext, we can use the toolkit in [47] to perform the ciphertext calculation during model training. However, this encryption strategy will lead to lower model training efficiency.

To protect the released model, we usually adopt the differential privacy to sanitize the trained model parameters. After that, the output of the trained model will be influenced and it is difficult for the attackers to infer the architecture and parameters of the trained model through the output of the model.

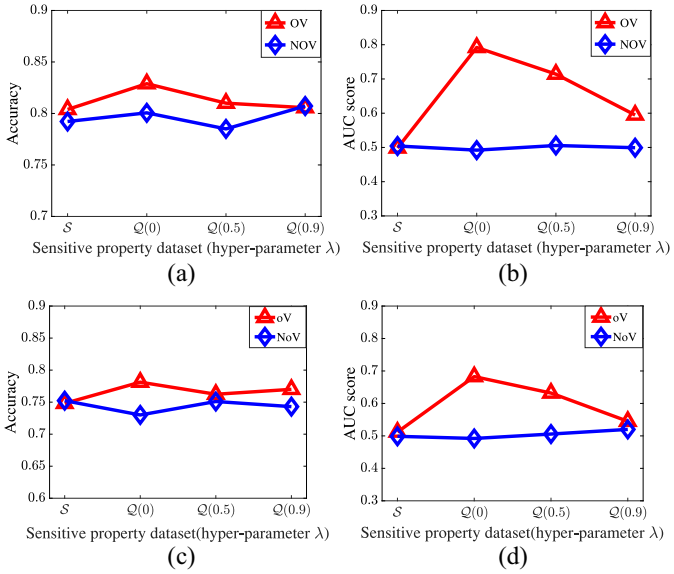


Fig. 6. Active inference against irrelevant properties. (a) LFW: Model accuracy. (b) LFW: Inference AUC score. (c) Adult: Model accuracy. (d) Adult: Inference AUC score.

VII. Performance Evaluation

In this section, we validate NOSnoop with three real-world data sets, labeled faces in the wild (LFW) [48], Facescrub [49], and Adult Income data set [50]. For the first two data sets, the model architecture is the same one as in [40] and [43], which is a VGGNet comprised by four convolutional layers. For the Adult data set, the model is designed as a four-layer neural network with batch normalization. We fix the meta-training to 10 000 epochs and the meta-adaption to 20 epochs, choose $N = 8$, $\alpha = 0.3$, and set MAML as a 2-way-5-shot learning. The experiments are conducted over a machine with a 2.6-GHz 32-core CPU, 128-GB RAM, and three TITAN V GPUs.

While evaluating the irrelevant properties, we choose gender (or income) as the main label and race as the sensitive property for LFW (or Adult). For the relevant property evaluation, we choose nose size in LFW as the main label and the sensitive property is also race. In the property inference attack, we assume the adversary can steal the specifical loss of victim (only victim and we use OV for short hereafter) or can reverse the aggregated loss calculated over several participants (not only victim and we utilize NOV for short hereafter). We represent the data set that has sensitive property data as $\mathbb{D}_{\text{sens}}$. If we put the sensitive property data only in $\mathcal{D}^s_{\text{train}}$, we denote $\mathbb{D}_{\text{sens}} = \mathcal{S}$ (for conciseness in the results). If the sensitive property data are evenly distributed in both $\mathcal{D}^s_{\text{train}}$ and $\mathcal{D}^q_{\text{train}}$, we denote $\mathbb{D}_{\text{sens}} = \mathcal{Q}$. When using the adversarial game with parameter λ, $\mathcal{Q}(\nu)$ means $\mathbb{D}_{\text{sens}} = \mathcal{Q}$ and $\lambda = \nu$. We use the AUC score to evaluate the performance of our defence. The AUC score close to 0.5 indicates that NOSnoop can defend against the property inference attack well.

A. Evaluation of Irrelevant Property Preservation

We first evaluate the privacy preservation of irrelevant properties. The simulation results of active inference attack are

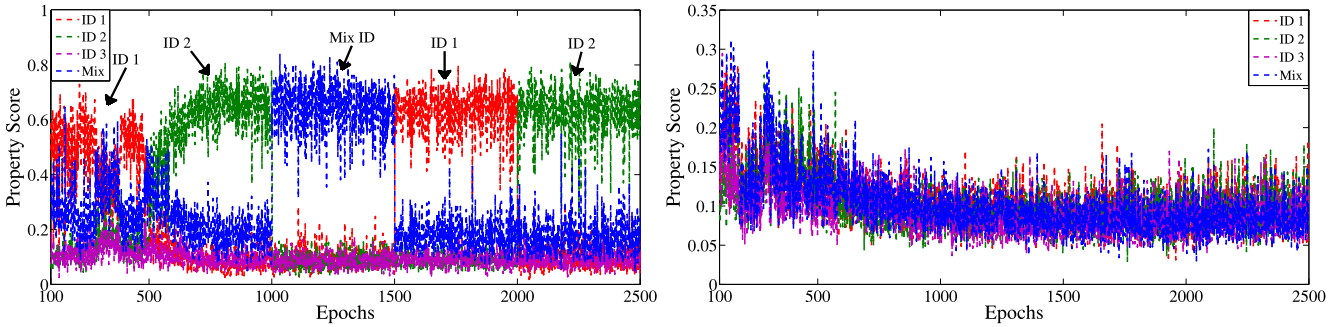


Fig. 7. Preserving the occurring sensitive property (best viewed for online color version). (a) Inference attack for identity occurrence. (b) Inference against occurring identity in $\mathcal{D}^s_{\text{train}}$.

TABLE III
INFERENCE AGAINST IRRELEVANT PROPERTIES ($\mathbb{D}_{\text{SENS}} = \mathcal{S}$)

Inference attack		Active	Passive
OV	Accuracy	0.8039	0.7922
	AUC Score	0.4984	0.4736
NOV	Accuracy	0.7878	0.7906
	AUC Score	0.5042	0.4987

shown in Fig. 6. We consider the adversary steals the empirical loss in two cases: 1) OV and 2) NOV. Since the sensitive property p is irrelevant with the main prediction label, we regard that p in $\mathcal{D}^s_{\text{train}}$ or $\mathcal{D}^q_{\text{train}}$ has no effect on the trained model accuracy. As shown in Fig. 6(a) and (c), the model accuracy does not have an obvious change when $\mathbb{D}_{\text{sens}} = \mathcal{S}$ or $\mathbb{D}_{\text{sens}} = \mathcal{Q}$. Although we vary the parameter λ in case of $\mathbb{D}_{\text{sens}} = \mathcal{Q}$, it just has a slight influence on the model accuracy. When $\mathbb{D}_{\text{sens}} = \mathcal{S}$, we find that the active property inference attack almost failed [AUC score ≈ 0.5, as shown in Fig. 6(b) and (d)]. But if $\mathbb{D}_{\text{sens}} = \mathcal{Q}$, the AUC score reaches 0.78 when $\lambda = 0$ in case of OV for LFW. It means the adversary has a high probability to infer p. Another interesting result is that if the victim has one or more honest participants to train the model collaboratively, the adversary also cannot infer the sensitive property [the case of NOV in Fig. 6(b) and (d)]. Hence, when preserving the sensitive property that is irrelevant with the main label, we can hide the sensitive property data in $\mathcal{D}^s_{\text{train}}$ without compromising the model accuracy.

We also evaluate the passive inference attack and compare the results with active inference attack in Table III. Since the active inference attack still cannot infer the property successfully, the passive inference attack also failed when we put the sensitive property data in $\mathcal{D}^s_{\text{train}}$ in case of OV or NOV. The active inference has a stronger ability to distinguish the feature representation than passive. So the passive attack also failed when we protect the irrelevant sensitive property in $\mathcal{D}^s_{\text{train}}$.

In [3], the adversary can infer a suddenly occurring property during model training. Therefore, we also evaluate NOSnoop to show that it can defend this attack. Similar to [3], the main prediction task is gender classification on Facescrub data set. The purpose of adversary is to infer whether and when a certain person appeared in the training batch. We fix the collaborative training to 2500 epochs and the adversary starts collecting the empirical loss from 100 epochs. We arrange two specific identities that appear in certain epochs: ID 1 appears in $0 \sim 500$ and $1500 \sim 2000$ epochs, ID 2 appears in $500 \sim 1000$ and $2000 \sim 2500$ epochs, and the mixture of ID 1 and ID 2 appears in $1000 \sim 1500$ epochs. As shown in Fig. 7(b), when we hide the occurring identity in meta-training support data set $\mathcal{D}^s_{\text{train}}$, the adversary cannot distinguish in which epoch the identity occurs. If we do not take such a privacy preservation measure, the adversary will easily infer the occurrence of identities, as shown in Fig. 7(a).

B. Evaluation of Relevant Property Preservation

To show the preservation of relevant properties, we evaluate NOSnoop on the LFW data set and set nose size as the main prediction label. In the case of OV, we find that the adversary obtains a high AUC score when we just put the sensitive property data in $\mathcal{D}^q_{\text{train}}$, as shown in Fig. 8(b). Thus, we introduce the adversarial game, which is described in Section V-B and discriminate the sensitive property by adjusting parameter λ. When we set $\lambda = 3$, we find that the inference AUC score is close to 0.5, which means the inference attack fails. It is also found that when $\lambda = 5$ and 10, the proposed scheme can still achieve good performance to defend the property inference attack, demonstrating the robustness of the proposed scheme. However, as the sensitive property is relevant with the main predicted label, it also influences the main label prediction when we conceal the sensitive property. As shown in Fig. 8(a), by varying λ, we find that the training accuracy of meta-model decreases slightly. But after retraining in the meta-adaptation process, participants' personalized model accuracy is almost not influenced and remains at 0.8. Although we can put the sensitive property data in $\mathcal{D}^s_{\text{train}}$ to obtain a low inference AUC score, it will greatly reduce the trained model accuracy, even after the retraining process [$\mathbb{D}_{\text{sens}} = \mathcal{S}$ in Fig. 8(a) and (b)].

In the case of NOV, the adversary cannot obtain the victim's empirical loss. Thus, it is difficult for the adversary to launch the inference attack successfully, which has been evaluated for the irrelevant property in Fig. 6. For the relevant property preservation, we also find similar results, as shown in Fig. 9. In addition, it may also be found that the proposed scheme's performance is stable and robust when $\lambda \geq 3$ in terms of accuracy and tackling property inference attack. Since

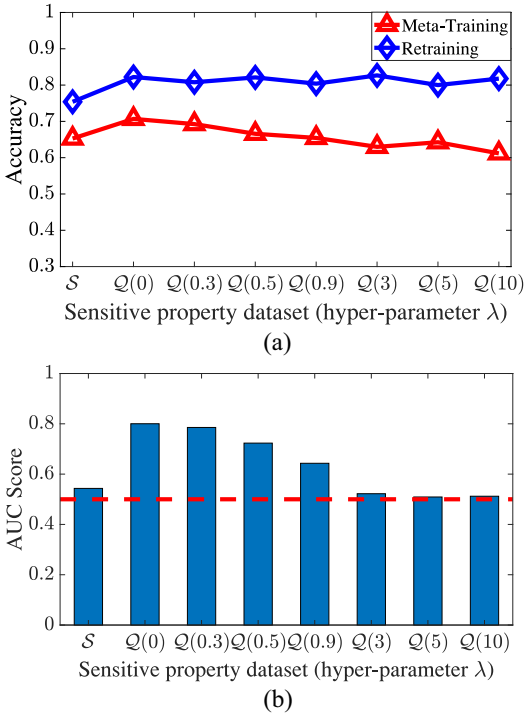


Fig. 8. Active inference (OV case) against relevant properties. (a) Model accuracy. (b) Inference AUC score.

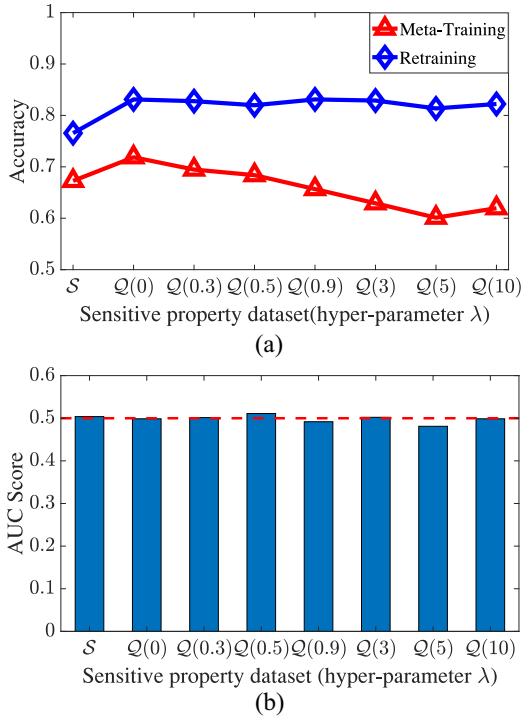


Fig. 9. Active inference (NOV case) against relevant properties. (a) Model accuracy. (b) Inference AUC score.

the adversary can only obtain the aggregated loss from several participants, it cannot infer the sensitive property even if we put the sensitive property data in $\mathcal{D}^q_{\text{train}}$ and set $\lambda = 0$. Hence, if one or more honest participants take part in the collaborative training, the victim can put the sensitive property data in $\mathcal{D}^q_{\text{train}}$ to ensure the training model accuracy without worrying about the property inference attack.

We show the model training process for different $\mathbb{D}_{\text{sens}}$ ($\mathcal{S}$, $\mathcal{Q}(0)$, $\mathcal{Q}(3)$) in Fig. 10. From the simulation results, we find that all three meta-training processes can quickly reach convergence as shown in Fig. 10(a)–(c). But the testing accuracies of meta-models are different. When we put the sensitive property data in $\mathcal{D}^s_{\text{train}}$ ($\mathbb{D}_{\text{sens}} = \mathcal{S}$), the testing accuracy of meta-model only reaches about 0.65. But if we put the relevant sensitive property data in $\mathcal{D}^q_{\text{train}}$ and set $\lambda = 0$, the testing accuracy reaches 0.7. The reason for this is that the victim shares the empirical loss calculated on the sensitive property data to help to update the meta-model. Since the features extracted from relevant property help to predict the main label, CS can obtain more useful information from the received empirical loss. If we set $\lambda = 3$ to reduce the features related to the sensitive property, CS misses some necessary information to update the meta-model. So the meta-training also obtains a low testing accuracy when we set $\mathbb{D}_{\text{sens}} = \mathcal{Q}(3)$. But after retraining in meta-adaptation, the model accuracies for $\mathbb{D}_{\text{sens}} = \mathcal{Q}(0)$ and $\mathcal{Q}(3)$ just have a little difference, as shown in Fig. 10(d). If more participants conceal their sensitive properties by adversarial game, this accuracy difference may be enlarged. But that is still acceptable. If participants hide the sensitive property data in $\mathcal{D}^s_{\text{train}}$, CS obtains nothing about the sensitive property to update the meta-model and this cannot be remedied. So

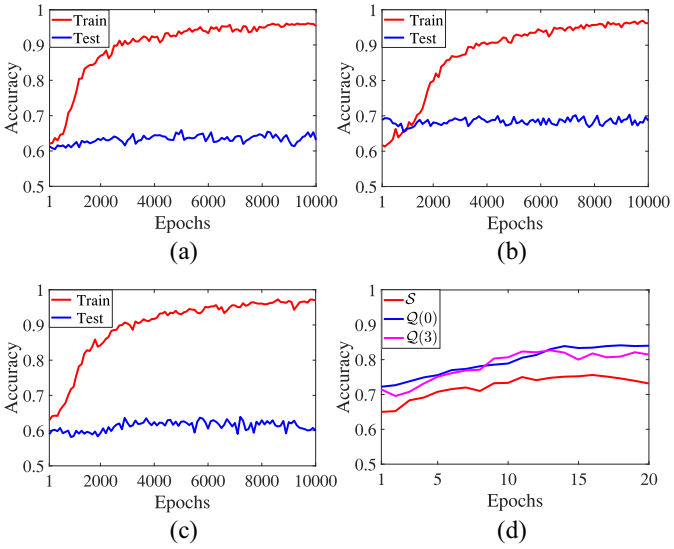


Fig. 10. Training accuracy for relevant properties. (a) $\mathbb{D}_{\text{sens}} = \mathcal{S}$. (b) $\mathbb{D}_{\text{sens}} = \mathcal{Q}(0)$. (c) $\mathbb{D}_{\text{sens}} = \mathcal{Q}(3)$. (d) Meta-adaption retrain accuracy.

the retraining model accuracy is still lower than the case of $\mathbb{D}_{\text{sens}} = \mathcal{Q}$.

Finally, we further compare NOSnoop with the NPP mechanism and DP mechanism in Fig. 11. Since the adversary launches the inference attack based on the collected gradients, we design a DP model using gradient perturbation and use the Rényi differential privacy [51] to analyze the privacy loss. We keep $\delta = 10^{-5}$ and vary privacy budget ϵ between 0.1 and 500. We find that the inference AUC score ≈ 0.5 when we set $\epsilon = 0.1$, 1, and 5. But the highest model accuracy is only 0.5285. If we want to improve the model utility by varying ϵ,

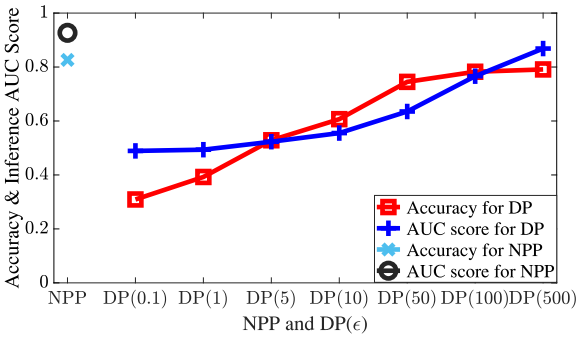


Fig. 11. Defense performance of NPP and DP.

the AUC score will also increase rapidly. It means the adversary launches the property inference attack successfully. Thus, we cannot obtain a tradeoff between model utility and the vulnerability of property inference attack. In NPP, we can obtain the highest model accuracy of 0.8264, but the inference AUC score reaches up to 0.9296, indicating a vulnerable system to inference attack.

VIII. Conclusion

In this article, we have proposed a novel mechanism, namely, NOSnoop, to defend against property inference attack in collaborative model training. Specifically, we considered two cases: 1) irrelevant property preservation and 2) relevant property preservation. For the irrelevant property preservation, we introduced the meta-learning and hid the sensitive property data in the meta-training support data set. Thus, the adversary could not infer the sensitive property and the model accuracy was not influenced. For the relevant property preservation, we needed to put the sensitive property data in meta-training query data set to ensure the model utility. Therefore, an adversarial game was constructed to conceal the features about sensitive property and we improved the model accuracy through model retraining in meta-adaptation process. We evaluated the effectiveness of NOSnoop and the results justified that NOSnoop was effective to defend the property inference attack during model training. As part of our future work, we will consider the inference attack with the black-box access to the trained model.

References

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

[2] M. Abadi *et al.*, "Deep learning with differential privacy," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2016, pp. 308–318.

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

[4] X. Ma, J. Ma, H. Li, Q. Jiang, and S. Gao, "PDLM: Privacy-preserving deep learning model on cloud with multiple keys," *IEEE Trans. Services Comput.*, vol. 14, no. 4, pp. 1251–1563, Aug. 2021.

[5] L. T. Phong, Y. Aono, T. Hayashi, L. Wang, and S. Moriai, "Privacy-preserving deep learning via additively homomorphic encryption," *IEEE Trans. Inf. Forensics Security*, vol. 13, pp. 1333–1345, 2018.

[6] S. Truex, L. Liu, M. E. Gursoy, W. Wei, and L. Yu, "Effects of differential privacy and data skewness on membership inference vulnerability," 2019. [Online]. Available: arXiv:1911.09777.

[7] F. Chen, Z. Dong, Z. Li, and X. He, "Federated meta-learning for recommendation," 2018. [Online]. Available: arXiv:1802.07876.

[8] S. Lin, G. Yang, and J. Zhang, "A collaborative learning framework via federated meta-learning," in *Proc. IEEE 40th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Singapore, 2020, pp. 289–299.

[9] S. Gao, J. Ma, W. Shi, G. Zhan, and C. Sun, "TrPF: A trajectory privacy-preserving framework for participatory sensing," *IEEE Trans. Inf. Forensics Security*, vol. 8, pp. 874–887, 2013.

[10] X. Ma *et al.*, "APPLET: A privacy-preserving framework for location-aware recommender system," *Sci. China Inf. Sci.*, vol. 60, no. 9, 2017, Art. no. 92101.

[11] X. Ma, J. Ma, H. Li, Q. Jiang, and S. Gao, "AGENT: An adaptive geo-indistinguishable mechanism for continuous location-based service," *Peer-to-Peer Netw. Appl.*, vol. 11, no. 3, pp. 473–485, 2018.

[12] K. Gu, L. Wang, and B. Yin, "Social community detection and message propagation scheme based on personal willingness in social network," *Soft Comput.*, vol. 23, no. 15, pp. 6267–6285, 2019.

[13] X. Ma, J. Ma, H. Li, Q. Jiang, and S. Gao, "ARMOR: A trust-based privacy-preserving framework for decentralized friend recommendation in online social networks," *Future Gener. Comput. Syst.*, vol. 79, pp. 82–94, Feb. 2018.

[14] Q. Jiang, N. Zhang, J. Ni, J. Ma, X. Ma, and K.-K. R. Choo, "Unified biometric privacy preserving three-factor authentication and key agreement for cloud-assisted autonomous vehicles," *IEEE Trans. Veh. Technol.*, vol. 69, no. 9, pp. 9390–9401, Sep. 2020.

[15] Q. Jiang, Z. Chen, J. Ma, X. Ma, J. Shen, and D. Wu, "Optimized fuzzy commitment based key agreement protocol for wireless body area network," *IEEE Trans. Emerg. Topics Comput.*, vol. 9, no. 2, pp. 839–853, Apr.–Jun. 2021.

[16] N. Papernot, P. D. McDaniel, A. Sinha, and M. P. Wellman, "SoK: Security and privacy in machine learning," in *Proc. IEEE Eur. Symp. Security Privacy (EuroS&P)*, London, U.K., 2018, pp. 399–414.

[17] N. Waheed, X. He, M. Ikram, M. Usman, S. S. Hashmi, and M. Usman, "Security and privacy in IoT using machine learning and blockchain: Threats & countermeasures," 2020. [Online]. Available: arXiv:2002.03488.

[18] C. Briggs, Z. Fan, and P. Andras, "A review of privacy-preserving federated learning for the Internet-of-Things," 2020. [Online]. Available: arXiv:2004.11794.

[19] W. Wei *et al.*, "A framework for evaluating client privacy leakages in federated learning," in *Proc. 25th Eur. Symp. Res. Comput. Security*, 2020, pp. 545–566.

[20] D. Chen, N. Yu, Y. Zhang, and M. Fritz, "GAN-leaks: A taxonomy of membership inference attacks against generative models," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2020, pp. 343–362.

[21] F. Mo, A. Borovykh, M. Malekzadeh, H. Haddadi, and S. Demetriou, "Layer-wise characterization of latent information leakage in federated learning," 2020. [Online]. Available: arXiv:2010.08762.

[22] H. Weng, J. Zhang, F. Xue, T. Wei, S. Ji, and Z. Zong, "Privacy leakage of real-world vertical federated learning," 2020. [Online]. Available: arXiv:2011.09290.

[23] Q. Wang, S. Hu, M. Du, J. Wang, and K. Ren, "Learning privately: Privacy-preserving canonical correlation analysis for cross-media retrieval," in *Proc. IEEE Conf. Comput. Commun.*, Atlanta, GA, USA, 2017, pp. 1–9.

[24] P. Mohassel and Y. Zhang, "SecureML: A system for scalable privacy-preserving machine learning," in *Proc. IEEE Symp. Security Privacy*, San Jose, CA, USA, 2017, pp. 19–38.

[25] S. Wagh, D. Gupta, and N. Chandran, "SecureNN: 3-party secure computation for neural network training," *Proc. Privacy Enhancing Technol.*, vol. 2019, no. 3, pp. 26–49, 2019.

[26] P. Li *et al.*, "Multi-key privacy-preserving deep learning in cloud computing," *Future Gener. Comput. Syst.*, vol. 74, pp. 76–85, Sep. 2017.

[27] J. Konecný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," 2016. [Online]. Available: arXiv:1610.05492.

[28] P. Kairouz *et al.*, "Advances and open problems in federated learning," 2019. [Online]. Available: arXiv:1912.04977.

[29] L. Xie, I. M. Baytas, K. Lin, and J. Zhou, "Privacy-preserving distributed multi-task learning with asynchronous updates," in *Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discovery Data Min.*, 2017, pp. 1195–1204.

[30] C. Zhou, A. Fu, S. Yu, W. Yang, H. Wang, and Y. Zhang, "Privacy-preserving federated learning in fog computing," *IEEE Internet Things J.*, vol. 7, no. 11, pp. 10782–10793, Nov. 2020.

[31] M. Malekzadeh, R. G. Clegg, and H. Haddadi, "Replacement AutoEncoder: A privacy-preserving algorithm for sensory data analysis," 2017. [Online]. Available: arXiv:1710.06564.

[32] M. Nasr, R. Shokri, and A. Houmansadr, "Machine learning with membership privacy using adversarial regularization," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2018, pp. 634–646.

[33] Z. Yang, B. Shao, B. Xuan, E.-C. Chang, and F. Zhang, "Defending model inversion and membership inference attacks via prediction purification," 2020. [Online]. Available: arXiv:2005.03915.

[34] M. Backes, P. Berrang, M. Humbert, and P. Manoharan, "Membership privacy in microRNA-based studies," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2016, pp. 319–330.

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

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

[37] D. Yu, H. Zhang, W. Chen, J. Yin, and T.-Y. Liu, "Membership inference with privately augmented data endorses the benign while suppresses the adversary," 2020. [Online]. Available: arXiv:2007.10567.

[38] S. Hidano, Y. Kawamoto, and T. Murakami, "TransMIA: Membership inference attacks using transfer shadow training," 2020. [Online]. Available: arXiv:2011.14661.

[39] X. Luo, Y. Wu, X. Xiao, and B. C. Ooi, "Feature inference attack on model predictions in vertical federated learning," 2020. [Online]. Available: arXiv:2010.10152.

[40] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proc. 34th Int. Conf. Mach. Learn. (ICML)*, vol. 70, 2017, pp. 1126–1135.

[41] F. Chen, M. Luo, Z. Dong, Z. Li, and X. He, "Federated meta-learning with fast convergence and efficient communication," 2018. [Online]. Available: arXiv:1802.07876.

[42] C. Fan and J. Huang, "Federated few-shot learning with adversarial learning," 2021. [Online]. Available: arXiv:2104.00365.

[43] A. Antoniou, H. Edwards, and A. J. Storkey, "How to train your MAML," in *Proc. 7th Int. Conf. Learn. Represent. (ICLR)*, 2019, pp. 1–11.

[44] S. Rezaei and X. Liu, "On the difficulty of membership inference attacks," 2020. [Online]. Available: arXiv:2005.13702.

[45] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in *Proc. Conf. Neural Inf. Process. Syst.*, 2014, pp. 3320–3328.

[46] Y. Ganin and V. S. Lempitsky, "Unsupervised domain adaptation by backpropagation," in *Proc. 32nd Int. Conf. Mach. Learn. (ICML)*, vol. 37, 2015, pp. 1180–1189.

[47] X. Liu, R. H. Deng, K.-K. R. Choo, and J. Weng, "An efficient privacy-preserving outsourced calculation toolkit with multiple keys," *IEEE Trans. Inf. Forensics Security*, vol. 11, pp. 2401–2414, 2016.

[48] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller, "Labeled faces in the wild: A database for studying face recognition in unconstrained environments," Dept. Comput. Sci., Univ. Massachusetts, Amherst, MA, USA, Rep. 07–49, Oct. 2007.

[49] H.-W. Ng and S. Winkler, "A data-driven approach to cleaning large face datasets," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Paris, France, 2014, pp. 343–347.

[50] A. Frank. (2010). *UCI Machine Learning Repository*. [Online]. Available: http://archive.ics.uci.edu/ml

[51] I. Mironov, "Rényi differential privacy," in *Proc. IEEE 30th Comput. Security Found. Symp. (CSF)*, Santa Barbara, CA, USA, 2017, pp. 263–275.

Xindi Ma received the B.S. degree in computer science and the Ph.D. degree in computer science from Xidian University, Xi'an, China, in 2013 and 2018, respectively.

He is currently a Postdoctoral with the School of Cyber Engineering, Xidian University. His current research interests include machine learning, location-based service, and recommender system with focus on security and privacy issues.

Baopu Li received the Ph.D. degree from the Chinese University of Hong Kong, Hong Kong, in 2008.

His current major research interests focus on machine learning and computer vision.

Qi Jiang received the B.S. degree in computer science from Shaanxi Normal University, Xi'an, China, in 2005, and the Ph.D. degree in computer science from Xidian University, Xi'an, in 2011.

He is currently a Professor with the School of Cyber Engineering, Xidian University. His research interests include security protocols and wireless network security, and cloud security.

Yimin (Ian) Chen received the B.S. degree from the School of Electronics Engineering and Computer Science, Peking University, Beijing, China, in 2010, the M.Phil. degree from the Department of Electrical Engineering, Chinese University of Hong Kong, Hong Kong, in 2013, and the Ph.D. degree from Arizona State University, Tempe, AZ, USA, in 2018.

His Ph.D. research focused on security and privacy in mobile computing. He is currently a Postdoctoral Associate with the Department of Computer Science, Virginia Tech, Blacksburg, VA, USA. His research interests encompass security and privacy in machine/meta-learning and networked systems.

Sheng Gao received the B.S. degree in information and computation science from Xi'an University of Posts and Telecommunications, Xi'an, China, in 2009, and the Ph.D. degree in computer science and technology from Xidian University, Xi'an, in 2014.

He is an Assistant Professor with the School of Information, Central University of Finance and Economics, Beijing, China. His current research interests include finance information security and privacy computing.

Jianfeng Ma (Member, IEEE) received the Ph.D. degree in computer science from Xidian University, Xi'an, China, in 1995.

He is currently a Professor with the School of Cyber Engineering, Xidian University. He has published over 150 journal and conference papers. His research interests include information security, cryptography, and network security.