Computers & Security 121 (2022) 102857

Contents lists available at ScienceDirect

Computers & Security

journal homepage: www.elsevier.com/locate/cose

# SEDML: Securely and efficiently harnessing distributed knowledge in machine learning

Yansong Gao[a], Qun Li[a], Yifeng Zheng[b,*], Guohong Wang[a], Jiannan Wei[a], Mang Su[a]

[a] School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing, JiangSu, China
[b] School of Computer Science and Technology, Harbin Institute of Technology, Shenzhen, Guangdong, China

A R T I C L E I N F O

*Article history:*
Received 15 October 2021
Revised 28 March 2022
Accepted 21 July 2022
Available online 23 July 2022

*Keywords:*
Distributed learning
Knowledge transfer
Privacy protection
Secure computation
Differential privacy

A B S T R A C T

Training high-performing machine learning models require a rich amount of data which is usually distributed among multiple data sources in practice. Simply centralizing these multi-sourced data for training would raise critical security and privacy concerns, and might be prohibited given the increasingly strict data regulations. To resolve the tension between privacy and data utilization in distributed learning, a machine learning framework called private aggregation of teacher ensembles (PATE) has been recently proposed. PATE harnesses the knowledge (label predictions for an unlabeled dataset) from distributed teacher models to train a student model, obviating access to distributed datasets. Despite being enticing, PATE does not offer protection for the individual label predictions from teacher models, which still entails privacy risks. In this paper, we propose SEDML, a new protocol which allows to securely and efficiently harness the distributed knowledge in machine learning. SEDML builds on lightweight cryptography and provides strong protection for the individual label predictions, as well as differential privacy guarantees on the aggregation results. Extensive evaluations show that while providing privacy protection, SEDML preserves the accuracy as in the plaintext baseline. Meanwhile, SEDML outperforms the state-of-the-art work of Xiang et al. (ICDCS'20) by 43× in computation and 1.23× in communication.

## 1. Introduction

Machine learning (ML) models have been employed in various applications including medical diagnosis, speech recognition, face recognition, and financial fraud detection (Bakator and Radosav, 2018; LeCun et al., 2015), given their unprecedented performance. Training ML models for a high accuracy performance relies on rich data, which is usually collected from multiple data sources and aggregated in a centralized data center. However, sharing the data directly to a single centralized party for training is not always possible because of severe privacy concerns, especially for sensitive data such as medical images and bank information. In addition, the data aggregator must pay great attention to the data regulations such as the General Data Protection Regulation (GDPR, effective from May, 2018) (Europe, 2016), California Privacy Rights Act (CPRA, effective from Jan, 2021) (States, 2020), and China Data Security Law (CDSL, effective from Sep 2021) (China, 2021). The aggregator could face severe legal issues whenever the collected data is misused or inappropriately processed.

One solution is to train the ML model over centralized but encrypted data. One representative work is by Mohassel and Zhang (2017), which allows secure model training over encrypted data for protecting data privacy. However, the substantial amount of data also introduces major challenges to ML such as high data dimensionality and model scalability (Chen and Lin, 2014; Najafabadi et al., 2015). A more pragmatic solution is distributed learning that does not need to access local data, which can significantly reduce privacy leakages while still harnessing the distributed isolated rich data (Gao et al., 2021). One popular paradigm is Federated Learning (FL) (Wang et al., 2019), where clients perform local training and only share model updates rather than raw data to the aggregator for updating a global model. Nonetheless, the model update parameters still expose notable information that can be exploited by an adversary to infer private client data through, e.g., membership inference attack (Rahman et al., 2018) and data inversion attack (Alves et al., 2019; Khosravy et al., 2021).

To further reduce the information communicated with the aggregation server in distributed learning, Papernot et al. (2017, 2018) proposed a new machine learning framework, namely Private Aggregation of Teacher Ensembles, or PATE for short. The

* Corresponding author.
*E-mail addresses:* yansong.gao@njust.edu.cn (Y. Gao), 120106222757@njust.edu.cn (Q. Li), yifeng.zheng@hit.edu.cn (Y. Zheng), wgh@njust.edu.cn (G. Wang), jnwei@njust.edu.cn (J. Wei), sumang@njust.edu.cn (M. Su).

https://doi.org/10.1016/j.cose.2022.102857
0167-4048/© 2022 Elsevier Ltd. All rights reserved.

PATE framework was first introduced in Papernot et al. (2017), and later formalized in Papernot et al. (2018), where new noisy aggregation mechanisms are presented to greatly improve the accuracy. It is an increasingly popular machine learning framework for harnessing distributed knowledge in various ML application and has received growing attention (e.g., Jordon et al., 2019a; Jordon et al., 2019b; Long et al., 2021; Wang et al., 2021; Xiang et al., 2020, to just list a few). Generally, the PATE framework harnesses the knowledge from distributed teacher models to train a student model (Papernot et al., 2018). The teacher models are locally trained by clients over their private datasets and utilized to produce label predictions for an unlabeled training dataset queried by a requester. The label predictions from multiple teacher models are aggregated by an aggregator or service provider, which are then returned to the requester for usage in training the student model. Hence, the whole process avoids access to the clients' local datasets. To mitigate the potential leakages from the aggregated labels revealed to the requester, differential privacy mechanisms are employed to add calibrated noises in the aggregation process. In this way, the PATE framework not only ensures the practicability of the model training, but also provides some privacy protection.

However, the direct expose of the individual label predictions to the aggregator could still leak private information about the teacher models or even the local datasets. To counter such confidentiality issue, Xiang et al. (2020) recently presented a homomorphic encryption-based design which leverages homomorphic encryption to encrypt the individual label predictions from the teacher models and employ two non-colluding servers to conduct aggregation in ciphertext domain. Their solution, however, is unsatisfactory due to the following limitations. Firstly, their design relies on expensive homomorphic encryption and poses heavy performance overheads. Secondly, their design requires the number of clients that will submit label predictions to be fixed in advance. So if any client fails to submit the label predictions later, their design will not correctly work out. Thirdly, the interactions between each client and the servers are not one-off in their design. In particular, after a subroutine of secure threshold check for a training example, all the clients and the servers need to interact again. So the clients should keep staying online until the secure threshold check has been performed for all the queried samples in the unlabeled training dataset. If any client fails to participate in the second round of interaction, their design cannot correctly proceed again.

In light of the above, in this paper, we propose a new design for securely and efficiently harnessing the distributed knowledge in machine learning. Our design follows the machine learning paradigm in PATE for knowledge transfer, yet provides strong protection for the confidentiality of individual label predictions throughout the aggregation procedure, as well as ensures differential privacy guarantee on the aggregated labels. Our design is customized for the PATE framework with delicate design considerations. In a broad sense our work is under the umbrella of privacy-preserving machine learning (PPML), which has attracted great attention in recent years. We note that the common blueprint followed by recent works on PPML is to delicately build on existing security techniques to design constructions tailored for specific PPML problems. For example, the work of Riazi et al. (2019) combines garbled circuits and binary neural networks to realize round-efficient online secure neural network inference. The work of Mishra et al. (2020) builds on neural architecture search (NAS) and cryptographic techniques (e.g., homomorphic encryption, secret sharing, and garbled circuits) to support secure neural network inference that allows much work to be pushed to a data-independent preprocessing phase. Our design follows the same blueprint and provides a new customized solution for securely and efficiently harnessing distributed knowledge under the PATE framework. We note that the techniques in SEDML can also possibly be adapted to other distributed machine learning frameworks. For example, the aggregation of model updates in federated learning (McMahan et al., 2017) requires weighted summation, which at a high level needs secure addition and multiplication upon the demand for encrypting the model updates for protection.

In comparison with the state-of-the-art work Xiang et al. (2020), our design presents the following differences. Firstly, our design only makes use of lightweight secret sharing techniques and is highly efficient. Secondly, our design does not require fixing the number of clients in advance for the subsequent secure aggregation process. Thirdly, our design does not require multiple rounds of interactions among the clients and servers, allowing the clients to just go offline after providing their encrypted label predictions. We summarize our main contributions below:

- We propose SEDML, a new protocol for securely and efficiently harnessing the distributed knowledge in machine learning. SEDML builds on lightweight cryptography and ensures strong protection for individual label predictions during the secure aggregation procedure, and differential privacy on the aggregated labels.
- We propose an efficient method in the secret sharing domain for secure identification of the highest (noisy) vote count during the secure aggregation procedure. This method is mainly based on secure extraction of the most significant bit in the secret sharing domain so as to allow efficient and secure comparison.
- We conduct a comprehensive performance evaluation on SEDML, in terms of computation, communication, and accuracy. Extensive results validate that the accuracy performance in SEDML is comparable to the plaintext baseline. Meanwhile, compared to the state-of-the-art security design (Xiang et al., 2020), SEDML takes $43\times$ less computation time and $1.23\times$ less communication.

The rest of the paper is organized as follows. Section 2 introduces some preliminaries. Section 3 gives the problem statement. Section 4 presents the detailed design of SEDML. Section 5 shows the experiment results. Section 6 concludes this paper.

## 2. Preliminaries

### 2.1. Knowledge transfer via aggregating teacher ensembles

The PATE framework proposed in Papernot et al. (2018) harnesses the knowledge from distributed teacher models to train the student model. This framework is advantageous in that the training of the student model does not need to access local datasets held by a set of clients. There are three components in the framework: teacher model, aggregation mechanism, and student model. Each teacher model is independently trained using a client's local dataset, which could be privacy-sensitive. The student model is trained under the supervision of teacher models by distilling knowledge from all teacher models based on an aggregation mechanism. In particular, it is assumed that the requester who wants to train the student model has access to a public but unlabeled dataset. Each teacher model produces a label prediction for each sample in the dataset. The label predictions on each example are then aggregated in plaintext domain through a dedicated mechanism with differential privacy guarantees, which produces a aggregated label. The student model is then trained on those samples labeled through the aggregation mechanism.

### 2.2. Additive secret sharing

Our design will rely on a lightweight cryptographic technique, additive secret sharing, to achieve a secure and efficient realization for aggregating teacher ensembles. In particular, we will make use of 2-of-2 additive secret sharing.

Given a value $\alpha \in \mathbb{Z}_{2^l}$, its 2-of-2 additive secret sharing is a pair $([\alpha]_0 = \alpha - r, [\alpha]_1 = r)$, where $r$ is a random value in $\mathbb{Z}_{2^l}$ and the subtraction is done in $\mathbb{Z}_{2^l}$ (i.e., result is modulo $2^l$). Each share reveals no information about the original value $\alpha$.

Suppose that two values $\alpha$ and $\beta$ are secret-shared among two parties $\mathcal{P}_0$ and $\mathcal{P}_1$, i.e., $\mathcal{P}_0$ holds $[\alpha]_0$ and $[\beta]_0$ while $\mathcal{P}_1$ holds $[\alpha]_1$ and $[\beta]_1$. The secret sharing $[\alpha \pm \beta]$ can be computed locally where each party $\mathcal{P}_i$ $(i \in \{0, 1\})$ directly computes $[\alpha \pm \beta]_i = [\alpha]_i \pm [\beta]_i$. Multiplication by a constant $\gamma$ on the value $\alpha$ can also be done locally, i.e., $[\alpha \cdot \gamma]_i = \gamma \cdot [\alpha]_i$. Multiplication over $[\alpha]$ and $[\beta]$ can be supported by using the Beaver's multiplication triple (Beaver, 1991; Corrigan-Gibbs and Boneh, 2017). That is, given the secret sharing of a multiplication triple $(t_1, t_2, t_3)$ where $t_3 = t_1 \cdot t_2$, $[\alpha \cdot \beta]$ can be obtained with one round of interaction between the two parties. In particular, each party $\mathcal{P}_i$ first computes $[e]_i = [\alpha]_i - [t_1]_i$ and $[f]_i = [\beta]_i - [t_2]_i$. Then, $\mathcal{P}_i$ broadcasts $[e]_i$ and $[f]_i$, and recovers $e$ and $f$. Lastly, $P_i$ computes $[\alpha \cdot \beta]_i = i \cdot e \times f + [t_1]_i \times f + [t_2]_i \times e + [t_3]_i$.

### 2.3. Differential Privacy

Differential privacy (Dwork et al., 2006) is a rigorous privacy notion which, intuitively, ensures that the output of aggregate statistics computation over a database is insensitive to changes in any data record.

**Definition 1. ($(\epsilon, \delta)$-differential privacy)** A randomized mechanism $\mathcal{M}$ with domain $\mathcal{D}$ and range $\mathcal{R}$ satisfies$(\epsilon, \delta)$-differential privacy if for any two adjacent inputs D, D′ $\in \mathcal{D}$ and for any output S $\subseteq \mathcal{R}$ it holds that:

$$Pr[\mathcal{M}(D) \in S] \leq e^{\epsilon} \cdot Pr[\mathcal{M}(D') \in S] + \delta \tag{1}$$

In the application of differential privacy to machine learning, adjacent inputs refer to two datasets that differ by one training sample. The randomized mechanism $\mathcal{M}$ is a training algorithm. The natural interpretation for the parameters $\epsilon$ and $\delta$ are as follows: $\epsilon$ represents the upper limit on the privacy loss, and $\delta$ represents the probability that the privacy guarantee may not hold.

Rényi Differential Privacy (RDP) (Mironov, 2017) generalizes pure differential privacy ($\delta = 0$), with the following advantages. Firstly, it has nice composition property. Secondly, it provides a cleaner way to capture the privacy guarantees of Gaussian noise used for $(\epsilon, \delta)$-differential privacy. The RDP mechanism is defined based on Rényi divergence, as stated below:

**Definition 2. (Rényi Divergence)**. The Rényi divergence between two distributions P and Q, with order $\alpha$ ($\alpha > 0$ and $\alpha \neq 1$), is defined as:

$$D_\alpha(P \parallel Q) \triangleq \frac{1}{\alpha - 1} \log \mathbb{E}_{x \backsim Q}\left[ (\frac{P(x)}{Q(x)})^{\alpha} \right]. \tag{2}$$

**Definition 3. (Rényi Differential Privacy)**. A randomized mechanism $\mathcal{M}$ guarantees $(\alpha, \epsilon)$-RDP with $\alpha \geq 1$ if for any neighboring datasets D and D′,

$$D_\alpha(\mathcal{M}(D) \parallel \mathcal{M}(D')) = \tag{3}$$

$$\frac{1}{\alpha - 1} log \mathbb{E}_{x \backsim \mathcal{M}(D)}\left[ (\frac{Pr(\mathcal{M}(D) = x)}{Pr(\mathcal{M}(D') = x)})^{\alpha - 1} \right] \leq \epsilon. \tag{4}$$

**Theorem 1.** *If a mechanism $\mathcal{M}$ consists of a sequence of adaptive mechanisms $\mathcal{M}_1$,..., $\mathcal{M}_k$ such that for any $i \in [k]$, $\mathcal{M}_i$ guarantees($\alpha$, $\epsilon_i$)-RDP, then $\mathcal{M}$ guarantees ($\alpha$, $\sum_{i=1}^{k} \epsilon_i$)-RDP.*

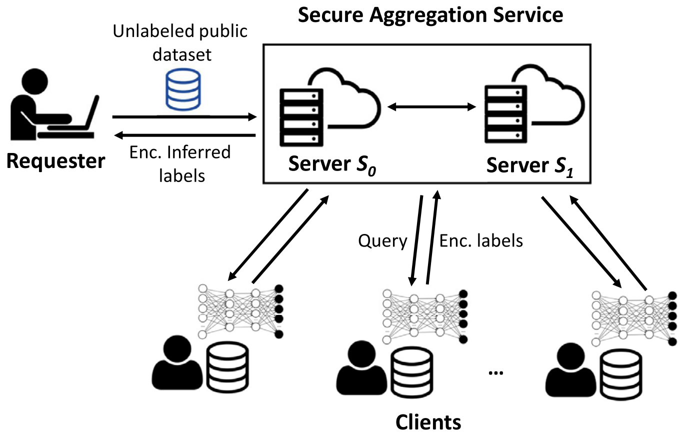

**Fig. 1.** The system architecture.

**Theorem 2.** ***(From RDP to DP)**. If a mechanism $\mathcal{M}$ guarantees ($\alpha$, $\epsilon$)-RDP, then $\mathcal{M}$ guarantees ($\epsilon + \frac{\log 1/\delta}{\alpha - 1}$, $\delta$)-differential privacy for any $\delta \in (0, 1)$.*

## 3. Problem statement

### 3.1. System architecture

Figure 1 illustrates the system architecture of SEDML that is aimed at securely and efficiently harnessing distributed knowledge in machine learning. At the core, there are three parties: the requester, clients, and the secure aggregation service provider. The requester wants to collect labels for a unlabeled public dataset via harnessing the collective knowledge of the clients, and then trains a model called student model. Each client holds a proprietary model, namely teacher model, which is trained on private datasets locally. On one hand, each client is interested in contributing knowledge to the training of the requester's student model via providing label predictions for the training examples in the public dataset through a teacher model trained over its local private dataset. On the other hand, each client also has privacy concerns regarding the label predictions provided for the public dataset as they may reveal information about its teacher model and thus the private dataset on which the teacher model has been trained. Hence, each client would only be willing to provide encrypted label predictions, and demand that security mechanisms should be put in place to safeguard their data privacy.

The secure aggregation service is a platform that bridges the requester and the clients. It could be deployed on the cloud given the well-known advantages like scalability, ubiquitous access, and economical cost. Similar to prior work Xiang et al. (2020), we consider that the secure aggregation service is jointly run by two cloud servers which are hosted by independent cloud providers. We note that such a two-server model has recently gained increasing traction in both academic work Riazi et al. (2018), Agrawal et al. (2019) and industrial sectors (Knott et al., 2020; Privacy, 2020).

In our system, the secure aggregation service receives label predictions in encrypted form from the clients, performs aggregation in the encrypted domain, and produces encrypted deferentially private aggregate label predictions for the public dataset, which are then returned to the requester on demand.

### 3.2. Threat assumptions and security goals

In SEDML, we consider threats primarily come from the two cloud servers providing the knowledge aggregation service, under the commonly assumed semi-honest adversary model. In particular, each cloud server will faithfully follows the protocol specifications of SEDML, yet may attempt to infer private sensitive information beyond their access rights, based on the messages received from the protocol execution. Here, following the state-of-the-art (Xiang et al., 2020) as well as other works Riazi et al. (2018), Agrawal et al. (2019), Privacy (2020), Knott et al. (2020), we assume the two cloud servers from different trust domains are non-colluding. The rationale behind such non-collusion assumption is that cloud providers are business-driven parties and usually well-established companies, so they have least incentives to risk their reputations by acting maliciously. With respect to the above threat model, our system aims to provide two following security guarantees:

1. **Confidentiality for individual label predictions.** The label predictions from individual clients are kept confidential throughout the service flow.
2. **Differential privacy for individual clients.** The aggregated label predictions revealed to the requester should be differentially private so that inferring private information about individual clients from the aggregated label predictions is thwarted.

## 4. The design of SEDML

### 4.1. Design rationale

To harness the distributed knowledge while being privacy-friendly, SEDML is aimed at securely aggregating the label predictions collected from the teacher models held by a set of clients, so that a student model can be trained by the requester based on the training examples with aggregated labels. We start with an overview of the aggregation mechanism (without considering differential privacy), which follows the plaintext-domain PATE framework (Papernot et al., 2018) and the state-of-the-art security design (Xiang et al., 2020). Without loss of generality, we describe the aggregation of label predictions for one data sample $x$ in the unlabeled public dataset for the sake of simplicity. Suppose there are $K$ clients, each of which holds a teacher model. We use $\mathbf{y}_j$ ($j \in \{1, \ldots, K\}$) to denote the label prediction from the $j$th teacher model for the training example $x$ in the unlabeled public dataset. The label prediction $\mathbf{y}_j$ from teacher model $j$ is an $N$-dimensional binary vector, given that there are $N$ classes in total. If the predicted class is the $i$th class, the $i$th element in the vector $\mathbf{y}_j$ – denoted by $\mathbf{y}_j(i)$ – is 1, and all other elements are 0.

We denote the vote count for the $i$th class as $n_i$, which is computed as $n_i = \sum_j \mathbf{y}_j(i)$. According to Papernot et al. (2018), Xiang et al. (2020), the aggregation of the label predictions, without considering differential privacy, works as follows. Firstly, the vote count $n_i$ for each class $i$ is computed, followed by the computation of the highest vote count $n^*$, i.e., $n^* = \max(n_1, \ldots, n_N)$. The highest vote count $n^*$ is then compared to a threshold $T$. If $n^* \geq T$, which means there is a consensus among the teacher models, the class $i^*$ corresponding to $n^*$ is output as the aggregated label for the training example $x$. Otherwise, a termination symbol $\perp$ is returned. Therefore, only the training examples with a consensus-reached aggregated class label will be used in training the student model.

**Challenges.** Although the above aggregation mechanism has no direct access to clients' local models and datasets, the label predictions from the teacher models can still pose a great threat to data privacy (Erdogan et al., 2021; Li and Zhang, 2020), which is overlooked in the PATE framework. Therefore, the aggregation mechanism should be performed while keeping the label predictions collected from the teacher models confidential. In addition, for the training examples which have a class whose highest votes are greater than the threshold, the aggregated label should be produced in the encrypted form as well and delivered to the requester on demand. For other training examples, they should be discarded as no consensus is reached among the teacher models.

That is, it is expected that throughout the whole workflow, the aggregation service *only learns whether there is a consensus among the teacher models given a training example in the dataset, and nothing beyond.*

From the above aggregation procedure, it is noted that the aggregation of the label predictions from the teacher models requires the atomic operations of addition and comparison. For securing the aggregation process, one may considering the use of homomorphic encryption as taken by the state-of-the-art work Xiang et al. (2020). However, homomorphic encryption is expensive and incurs significant performance overheads.

**Our Approach.** In SEDML, to ensure security while ensuring high efficiency, we resort to the lightweight technique of additive secret sharing for data encryption and processing, in contrast to the expensive homomorphic encryption used in Xiang et al. (2020). Despite that we note that addition in the additive secret sharing domain can be directly supported, as shown in the preliminaries (Section 2.2). There is a dearth of efficiently supporting secure comparison in the additive secret sharing domain. Our observation is that secure comparison of two values $x$ and $y$ in the additive secret sharing domain can be realized via securely extracting the most significant bit of the subtraction result $x - y$ between two values in the ring $\mathbb{Z}_{2^l}$ (Liu et al., 2021; Zheng et al., 2019). We further observe that the MSB extraction can be ingeniously achieved via implementing a full adder logic in the secret sharing domain.

Inspired by prior work Liu et al. (2021), we take advantage of the carry look-ahead adder for realizing secure and efficient comparison in SEDML, considering the fact that it obviates the cumbersome sequential carry computation and thus consumes much less number of rounds, in contrast to the standard ripple carry adder. It follows two general steps as below.

1. Firstly, a carry generate signal $G_i$ and a carry propagate signal $P_i$ are defined, which can be computed instantly based on the input bits $\{a_i\}$ and $\{b_i\}$, i.e., $G_i = a_i \cdot b_i$ and $P_i = a_i + b_i$.
2. Secondly, the carry bit computation can be formulated as $c_{i+1} = G_i + P_i \cdot c_i$. Such formulation allows a carry to be computed without waiting for the carry to ripple through all previous phases. Let us take a 4-bit carry look-ahead adder as an example. We have $c_4 = G_3 + P_3 \cdot c_3 = G_3 + P_3 \cdot (G_2 + P_2 \cdot (G_1 + P_1 \cdot G_1))$.

With the above formulation, the MSB of a secret $l$-bit value with shares $\{a_i\}_0^{l-1}$ and $\{b_i\}_0^{l-1}$ in bitwise form can be securely obtained via computing $a_{l-1} + b_{l-1} + c_{l-1}$ in the secret sharing domain. Hence, given such secure MSB extraction, we are able to achieve efficient secure comparison in the secret sharing domain, as opposed to the prior design (Xiang et al., 2020) that relies on the expensive homomorphic encryption. However, there is another subtle challenge to be addressed specific to the secure label aggregation procedure. In particular, during the comparison procedure, the relationship between the vote counts should not be revealed.

To solve this issue, we propose to have the following efficient secure aggregation design.

Given the secret sharings $[a]$ and $[b]$ of two values $a$ and $b$, we first get the secret-shared comparison result $[e]$ underlying which the plaintext value $e$ is either 0 ($a \geq b$) or 1 ($a < b$). Then, to achieve oblivious selection of the greater value, we compute the following: $[d] = [1 - e] \cdot [a] + [e] \cdot [b]$, which is the secret sharing of the greater value. If $e = 1$, $d = b$; otherwise $e = a$. In this manner, we can hide the relationship among the vote counts throughout the comparison procedure in secure aggregation of individual label predictions.

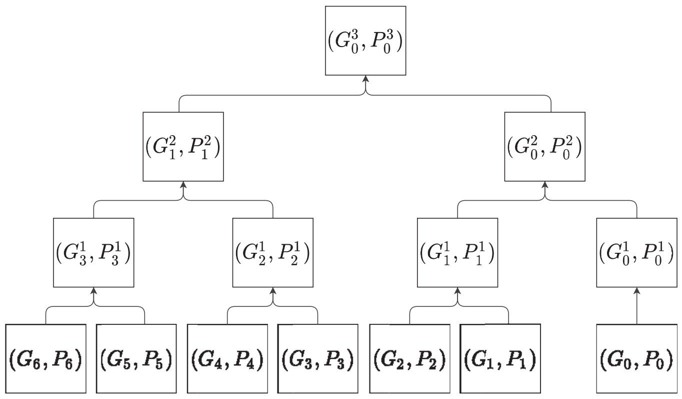

**Fig. 2.** Example of carry calculation over 8-bit inputs with the carry look-ahead adder.

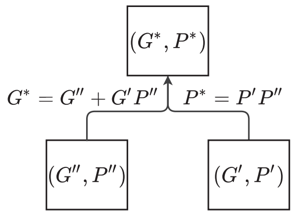

**Fig. 3.** Binary operator for carry calculation.

Given the secret sharing $[n^*]$ of the greatest vote count $n^*$, we can then perform a secure comparison with the threshold $T$ to indicate whether there is a consensus among the teacher models. To this end, the only information revealed to the cloud servers is whether there is consensus among the teacher models for each training example, and nothing beyond. To ensure differential privacy on the aggregated labels, we further follow Papernot et al. (2018), Xiang et al. (2020) and properly add Gaussian noises in the secure aggregation procedure.

### 4.2. The secure comparison gadget

Before elaborating on the SEDML protocol, we introduce the secure comparison gadget based on MSB extraction in the secret sharing domain, as mentioned above. The secure comparison gadget takes as input the secret sharings of two values $a$ and $b$, and outputs the secret sharing $[e]$ of the comparison result $e$. We note that the whole computation procedure for the carry look-ahead adder can be organized in the form of a binary tree, where the bottom layer consists of the signals $G$ and $P$ corresponding to the input bits. As an example, Fig. 2 illustrates the computation for the case of an 8-bit adder, where $G_0^3$ refers to the desired carry bit for the MSB computation. Let us define an operator $\circ$ to be used during the computation that illustrated in Fig. 3. Let $[\![\cdot]\!]$ denote secret sharing in the ring $\mathbb{Z}_2$, as opposed to secret sharing $[\cdot]$ in the ring $\mathbb{Z}_{2^l}$. Given the pre-generated multiplication triples in $\mathbb{Z}_2$ and $\mathbb{Z}_{2^l}$, the gadget $\mathsf{SCMP}_{ss}([a], [b]) \to [e]$ proceeds as the follows:

1. Each cloud server $S_i$ computes $[f] = [a] - [b]$.
2. Let $x_{l-1}, \ldots, x_0$ denote the bits for the share $[f]_0$ and $y_{l-1}, \ldots, y_0$ for the bits of the share $[f]_1$. Also, for $j \in [0, l-1]$, $S_0$ sets $[\![x_j]\!]_0 = x_j$ and $[\![y_j]\!]_0 = 0$; and $S_1$ sets $[\![x_j]\!]_1 = 0$ and $[\![y_j]\!]_1 = y_j$. For $j \in [0, l-1]$, $S_0$ sets $[\![d_j]\!]_0 = x_j$, and $S_1$ sets $[\![d_j]\!]_1 = y_j$.
3. $S_0$ and $S_1$ compute $[\![G_j]\!] = [\![x_j]\!] \cdot [\![y_j]\!]$ and $[\![P_j]\!] = [\![x_j]\!] + [\![y_j]\!]$, for $j \in [0, l-1]$.
4. $S_0$ and $S_1$ set $([\![G_0^1]\!], [\![P_0^1]\!]) = ([\![G_0]\!], [\![P_0]\!])$.
5. $S_0$ and $S_1$ proceed through the following rounds to securely compute the MSB, i.e., the secure comparison result.
   (a) In round $t = 1$, for $k \in [1, l/2 - 1]$, $S_0$ and $S_1$ compute $([\![G_k^1]\!], [\![P_k^1]\!]) = ([\![G_{2k}]\!], [\![P_{2k}]\!]) \circ ([\![G_{2k-1}]\!], [\![P_{2k-1}]\!]) = ([\![G_{2k}]\!] + [\![G_{2k-1}]\!] \cdot [\![P_{2k}]\!], [\![P_{2k-1}]\!] \cdot [\![P_{2k}]\!])$.
   (b) In each round $t \in [2, \log l - 1]$, for $k \in [0, l/2^t - 1]$, $S_0$ and $S_1$ compute $([\![G_k^t]\!], [\![P_k^t]\!]) = ([\![G_{2k+1}^{t-1}]\!], [\![P_{2k+1}^{t-1}]\!]) \circ ([\![G_{2k}^{t-1}]\!], [\![P_{2k}^{t-1}]\!]) = ([\![G_{2k+1}^{t-1}]\!] + [\![G_{2k}^{t-1}]\!] \cdot [\![P_{2k+1}^{t-1}]\!], [\![P_{2k}^{t-1}]\!] \cdot [\![P_{2k+1}^{t-1}]\!])$.
   (c) In round $t = \log l$, $S_0$ and $S_1$ compute $[\![G_0^t]\!] = [\![G_1^{t-1}]\!] + [\![G_0^{t-1}]\!] \cdot [\![P_1^{t-1}]\!] = [\![c_{l-1}]\!]$.
   (d) $S_0$ and $S_1$ compute $[\![e]\!] = [\![d_{l-1}]\!] + [\![c_{l-1}]\!]$.
6. $S_0$ and $S_1$ convert $[\![e]\!]$ in $\mathbb{Z}_2$ to $\mathbb{Z}_{2^l}$ as follows. $S_0$ sets $[p_1]_0 = [\![e]\!]_0$ and $[p_2]_0 = 0$, and $S_1$ sets $[p_1]_1 = 0$ and $[p_2]_1 = [\![e]\!]_1$. Then, $S_0$ and $S_1$ compute $[e] = [p_1] + [p_2] - 2[p_1][p_2]$.

From the above, we can see that the secure comparison gadget takes $O(\log l)$ communication rounds. Meanwhile, the procedure is fully conducted in the secret sharing domain with efficient arithmetic operations, with secret-shared inputs and output.

### 4.3. The SEDML protocol

We now present the complete SEDML protocol that allows to securely and efficiently harness distributed knowledge in machine learning. It builds on additive secret sharing to perform secure aggregation of the individual label predictions for training examples in the public dataset provided by the requester. It also provides strong assurance of differential privacy for clients engaged in the service. The complete SEDML protocol is shown in Algorithm 1, which is introduced below.

Given a training example $x$, each client $j$ produces a label prediction encoded as a binary vector $\mathbf{y}_j$, as introduced above. For privacy protection, client $j$ encrypts the vector $\mathbf{y}_j$ under additive secret sharing. In particular, client $j$ generates a vector $\mathbf{r}$ of random values sampled from $\mathbb{Z}_{2^l}$, and generates the shares $[\mathbf{y}_j]_0 = \mathbf{r}$ and $[\mathbf{y}_j]_1 = \mathbf{y}_j - \mathbf{r}$ through element-wise computation in the ring $\mathbb{Z}_{2^l}$. Client $j$ then sends the share $[\mathbf{y}_j]_0$ to cloud server $S_0$ and the share $[\mathbf{y}_j]_1$ to cloud server $S_1$ respectively. Upon receiving the secret shares $[\mathbf{y}]$ of the label predictions from the clients for a training example, the cloud servers perform aggregation over the secret shares to produce an aggregated label for the training example if there is a consensus among the teacher models, or terminate on that example otherwise.

The secure aggregation procedure works as follows. Firstly, leveraging the additive property of secret sharing, the cloud servers sum up the secret-shared label prediction vectors $\{[\mathbf{y}_j]\}_{j=1}^K$ and produce $[\mathbf{n}] = \sum_j [\mathbf{y_j}]$, which corresponds to the secret sharing of the votes for the classes. Then, the cloud servers need to obtain the encrypted highest vote count $n^*$ among the votes. By invoking the secure comparison gadget, the cloud servers can securely compare a pair of elements $[\mathbf{n}(p)]$ and $[\mathbf{n}(q)]$ in the secret-shared vector $[\mathbf{n}]$. That is, we have $\mathsf{SCMP}_{ss}([\mathbf{n}(p)], [\mathbf{n}(q)]) \to [e]$, where $[e]$ indicates the comparison result. Note that the secret-shared triples needed in the secure comparison gadget can be pre-generated offline and distributed to the two cloud servers by the requester. To obtain the secret sharing of the greater element, the cloud servers compute $[d] = [1 - e] \cdot [\mathbf{n}(p)] + [e] \cdot [\mathbf{n}(q)]$. Applying such secure comparison procedure over the votes in the secret-shared vector $\mathbf{n}$, the cloud servers can produce the secret-shared highest vote count $[n^*]$.

To check whether there is a consensus among the teacher models for the training example $x$, the cloud servers proceed as follows. Firstly, the cloud server $S_0$ adds a Gaussian noise $g \leftarrow \mathcal{N}(0, \sigma_1^2)$ to its share $[n^*]_0$, which leads to that the cloud servers now hold the secret sharing of the noisy highest vote count, i.e., $[n^* + g]$. Here, $\mathcal{N}(0, \sigma_1^2)$ means that the Gaussian distribution with mean

**Algorithm 1** The Proposed SEDML Protocol.

```
Input: Individual label prediction vectors $\{\mathbf{y}_j\}$.
Output: The aggregated label $i^*$ if there is a consensus among the teacher models, or $\perp$ otherwise.
    Client: // Encrypt the prediction vector.
 1: for each client $j$ do
 2:     Generate a vector of random values $\mathbf{r} \in \mathbb{Z}_{2^l}$ and set the secret shares as $[\mathbf{y}_j]_0 = \mathbf{r}$ and $[\mathbf{y}_j]_1 = \mathbf{y}_j - \mathbf{r}$.
 3:     Send the share $[\mathbf{y}_j]_0$ to cloud server $S_0$ and the share $[\mathbf{y}_j]_1$ to cloud server $S_1$ respectively.
 4: end for
    Cloud servers $S_0$ and $S_1$: // Phase 1: Secure Highest Vote Identification
 5: Sum up $\{[\mathbf{y}_j]\}_{j=1}^K$ and produce $[\mathbf{n}] = \sum_j [\mathbf{y_j}]$.
 6: $[n^*] = [\mathbf{n}(0)]$.
 7: for $(i = 1; i < N; i++)$ do
 8:     $\mathsf{SCMP}_{ss}([n^*], [\mathbf{n}(i)]) \to [e]$.
 9:     $[n^*] = [1 - e] \cdot [n^*] + [e] \cdot [\mathbf{n}(i)]$.
10: end for
11: Produce the secret-shared highest vote count $[n^*]$.
    Cloud servers $S_0$ and $S_1$:// Phase 2: Secure Threshold Check.
12: $S_0$ adds $g \leftarrow \mathcal{N}(0, \sigma_1^2)$ to its share $[n^*]_0$ and produces $[n^* + g]_0 = [n^*]_0 + g$.
13: $S_1$ sets $[n^* + g]_1 = [n^*]_1$.
14: Invoke $\mathsf{SCMP}_{ss}([n^* + g], [T]) \to [t]$ and reconstruct $t$;
15: if $t = 1$ then
16:     return $\perp$;
17: else
18:     go to next phase;
19: end if
    Cloud servers $S_0$ and $S_1$: // Phase 3: Secure Consensus Label Identification $[i^*]$.
20: for $(i = 0; i < N; i++)$ do
21:     $S_0$ samples $g_i \leftarrow \mathcal{N}(0, \sigma_2^2)$.
22:     $S_0$ computes $[\mathbf{m}(i)]_0 = [\mathbf{n}(i)]_0 + g_i$.
23:     $S_1$ sets $[\mathbf{m}(i)]_1 = [\mathbf{n}(i)]_1$.
24: end for
25: $[m^*] = [\mathbf{m}(0)]$.
26: $[s] = [0]$
27: for $(i = 1; i < N; i++)$ do
28:     $\mathsf{SCMP}_{ss}([m^*], [\mathbf{m}(i)]) \to [e]$.
29:     $[m^*] = [1 - e] \cdot [m^*] + [e] \cdot [\mathbf{m}(i)]$.
30:     $[s] = [1 - e] \cdot [s] + [e] \cdot [i]$.
31: end for
32: Set $[i^*] = [s]$ and send $[i^*]$ to the requester upon request.
    Requester: // Reconstruct the aggregated label $i^*$.
33: $i^* = [i^*]_0 + [i^*]_1$.
```

0 and variance $\sigma_1^2$. Note that addition of Gaussian noise is due to the demand for differential privacy. Then, the cloud servers invoke the secure comparison gadget which takes as input the secret sharings of the noisy highest vote $[n^* + g]$ and the threshold $[T]$. That is, we have $\mathsf{SCMP}_{ss}([n^* + g], [T]) \to [t]$. The cloud servers then reconstruct $t$ by exchanging the shares of $t$. If $t = 1$, we have $n^* + g < T$, so there is no consensus among the teacher models and the cloud servers terminate on the training example x. In such case, the training example $x$ is discarded and will not be used by the requester when training the student model. If $t = 0$, we have $n^* + g \geq T$, so there is a consensus among the teacher models.

The cloud servers now proceed to produce the secret-shared aggregated label for $x$. Firstly, the cloud server $S_0$ adds a Gaussian noise to each element of the vector $\mathbf{n}$ in the secret sharing domain, producing a secret-shared noisy vector $\mathbf{m}$. In particular, for each element $\mathbf{n}(i)$, the cloud server $S_0$ samples a noise $g_i$ from the Gaussian distribution $\mathcal{N}(0, \sigma_2^2)$ and computes $[\mathbf{m}(i)]_0 = [\mathbf{n}(i)]_0 + g_i$. The cloud server $S_1$ sets $[\mathbf{m}(i)]_1 = [\mathbf{n}(i)]_1$. In such way, the secret sharing of the vector of noisy vote counts is generated. The cloud servers then invoke the secure comparison gadget over the vector $[\mathbf{m}]$. Here, it is noted that in the end the cloud servers need to identify the index $i^*$ of the greatest value in the vector $\mathbf{m}$ after the secure comparison procedure. Therefore, while securely comparing two elements $\mathbf{m}(p)$ and $\mathbf{m}(q)$ of the vector $\mathbf{m}$ in the secret sharing domain, the cloud servers generate the secret sharing of the index $s$ of the greater value among them. In particular, given that $\mathsf{SCMP}_{ss}([\mathbf{m}(p)], [\mathbf{m}(q)]) \to [z]$, the cloud servers compute $[s] = [1 - z] \cdot p + [z] \cdot q$. It is easy to see that if $\mathbf{m}(p) \geq \mathbf{m}(q)$, we have $z = 0$, so $s = p$; and otherwise $s = q$. Applying such comparison procedure, the cloud servers finally obtain the secret-shared index $[i^*]$ of the highest noisy vote count in $\mathbf{m}$, which corresponds to the aggregated label for the training example $x$. This secret sharing can be delivered to the requester on demand, from which the requester can recover the aggregated label $i^* = [i^*]_0 + [i^*]_1$ for the training example $x$ and use it in training the student model.

**Remarks.** It is noted that directly comparing the vote counts sequentially requires $O(m)$ rounds of interactions among the cloud servers. Although this is already a linear increased complexity, it may still be a bottleneck, especially when the system runs in high-latency networks. To counter this, specific interaction reductions can be applied in the process of secure comparison of the elements of the secret vector for further efficacy optimization. In particular, we can partition the vote counts into groups with a size 2. Then, secure comparison can performed for the two values within each group in parallel, meaning that the communication can be batched. The (secret-shared) greater values from the secure comparison in each group form new groups for the next round of computation. In the end, the secret-shared greatest vote count is produced.

Our SEDML protocol fully runs in the secret sharing domain without heavy cryptography, as opposed to the state-of-the-art design (Xiang et al., 2020) that relies on expensive homomorphic encryption. We also note that the design of Xiang et al. (2020) needs multi-round communication among the clients and the cloud servers, while the clients in SEDML can just go offline after submitting their encrypted label predictions. Furthermore, we note that the design of Xiang et al. (2020) requires the number of participating clients to be determined and fixed in the beginning. All clients are required to participate subsequently, and their design will fail even if one of the clients fails to participate and submit ciphertexts. Our SEDML protocol is free of such practical restriction.

## 4.4. Security analysis

The SEDML protocol provides assurance on confidentiality of the individual label predictions from clients, as well as differential privacy guarantees for clients. In particular, throughout the secure aggregation procedure, the cloud servers only learn whether there is a consensus among the teacher models for a training example, without learning the individual label predictions. Furthermore, the aggregation results provide differential privacy guarantees, which prevents information leakage by inference on the aggregation results. As the confidentiality is ensured by the use of cryptographic techniques, we prove such guarantee following the standard simulation-based paradigm. We start with giving the ideal functionality.

**Definition 4.** The ideal functionality $\mathcal{F}$ of securely harnessing distributed knowledge in SEDML is modeled as follows. Given a training example $x$, each client $j$ provides a label prediction vector $\mathbf{y}_j$ to $\mathcal{F}$. The requester and the two cloud servers input nothing to $\mathcal{F}$. Upon receiving $\{\mathbf{y}_j\}_{j=1}^K$ from the clients, $\mathcal{F}$ conducts aggregation.

If there is a consensus among the teacher models, $\mathcal{F}$ outputs an aggregated label to the requester. Otherwise, $\mathcal{F}$ returns nothing.

**Definition 5.** A protocol $\Pi$ securely realizes $\mathcal{F}$ if it provides the following guarantees. We require that a corrupted and semi-honest cloud server $S_i$ ($i \in \{0, 1\}$) leans no information about individual label predictions and the aggregated label. Formally, a probabilistic polynomial time (PPT) simulator $\mathsf{Sim}_{S_i}$ should exist and generate a simulated view $\mathsf{View}_{\mathsf{Sim}_{S_i}}$ for $S_i$ such that $\mathsf{View}_{\mathsf{Sim}_{S_i}}$ is indistinguishable to the view $\mathsf{View}^{\Pi}_{S_i}$ of $S_i$ in the real protocol execution, i.e., $\mathsf{View}^{\Pi}_{S_i} \overset{c}{\approx} \mathsf{View}_{\mathsf{Sim}_{S_i}}$.

**Theorem 3.** *Our SEDML protocol securely realizes the functionality $\mathcal{F}$ according to Definition 5, given that the two cloud servers are semi-honest adversaries and non-colluding.*

**Proof.** According to our security definitions, we need to show the existence of a simulator for either of the cloud servers. In the SEDML protocol, the roles of the two cloud servers are symmetric, so it is sufficient to show a simulator $\mathsf{Sim}_{S_0}$ for the cloud server $S_0$. Recall that the cloud server $S_0$ receives secret shares of label predictions in the very beginning and then works over the secret shares throughout the whole secure aggregation procedure, with interactions with the other cloud server $S_1$. Attributing to the security of additive secret sharing, the secret shares received by $S_0$ are uniformly random and can be easily simulated by the simulator $\mathsf{Sim}_{S_0}$ using random values.

During the computation of secure aggregation, the interactions among the cloud servers are to securely compare the (noisy) votes, based on the secure comparison gadget SCMP. According to the construction of SCMP, it takes as input secret-shared values and outputs secret-shared values as well, and the inner processing is secure addition and secure multiplication based on standard Beaver's triples in the secret sharing domain. Assume the simulator for the standard triple-based secure multiplication is $\mathsf{Sim}^B$. The simulator $\mathsf{Sim}_{S_0}$ can invoke $\mathsf{Sim}^B$ on random values for each interaction with the cloud server $S_0$. The security of Beaver's triple trick ensures that the view simulated by $\mathsf{Sim}^B$ is indistinguishable from the view of the cloud server $S_0$ in every secure multiplication in the real execution. The simulator $\mathsf{Sim}_{S_0}$ combines in order the view simulated by $\mathsf{Sim}^B$ on every secure multiplication, which are then used as its simulated view for the secure comparison gadget. Recall that during the computation, there is a secure comparison step where the cloud servers securely compare the secret-shard highest (noisy) vote with a threshold to see if there is a consensus among the teacher models, and the result is revealed to them, i.e., the comparison result $t$. For this step, $\mathsf{Sim}_{S_0}$ adjusts the honest server's share of $t$ such that the recovered value is indeed the consensus-checking result $t$. This concludes the simulation in our SEDML protocol. □

Following prior works Papernot et al. (2018), Xiang et al. (2020), our SEDML protocol also adds differential privacy noises so as to prevent information leakage from the aggregated results. We have the following theorem regarding the differential privacy guarantee.

**Theorem 4.** *The SEDML protocol provides $(\epsilon, \delta)$-differential privacy, where $\epsilon = \sqrt{2(9/\sigma_1^2 + 2/\sigma_2^2)\log 1/\delta} + (9/2\sigma_1^2 + 1/\sigma_2^2)$ and $\delta \in (0, 1)$.*

**Proof.** As our SEDML protocol applies differential privacy in the same way as the prior work Xiang et al. (2020), the proof is similar to Xiang et al. (2020). So we only give the main points here and omit the details. Specifically, the differential privacy mechanisms involved in the SEDML protocol consists of the sparse vector technique and the report-noisy-maximum technique, which correspond to steps 5 to 19 and steps 20 to 32 in Algorithm 1 respectively. According to Xiang et al. (2020), the sparse vector technique satisfies $(\alpha, 9\alpha/2\sigma_1^2)$-RDP, and the report-noisy-maximum technique satisfies $(\alpha, \alpha/\sigma_2^2)$-RDP. So given the composition property of RDP in Theorem 1, the SEDML protocol satisfies $(\alpha, 9\alpha/2\sigma_1^2 + \alpha/\sigma_2^2)$-RDP. By Theorem 2, we have $(\epsilon, \delta)$-differential privacy for the SEDML protocol, where $\epsilon \geq \sqrt{2(9/\sigma_1^2 + 2/\sigma_2^2)\log 1/\delta} + (9/2\sigma_1^2 + 1/\sigma_2^2)$ Xiang et al. (2020). □

## 5. Experiments

### 5.1. Setup

To validate the performance of our SEDML design, two popular datasets consisting of SVHN and MNIST are used for comprehensive experiments. All these two datasets have been used in closely-related works Papernot et al. (2018), Xiang et al. (2020).

The SVHN (street view house number) dataset contains images of house numbers as seen from Google Street View images (Netzer et al., 2011). Each image contains a set of Arabic numbers from '0' to '9'. Each colorful image sample has a size of $32 \times 32 \times 3$. The training set has 73,257 images, and the testing set contains 26,032 images and 531,131 additional images–all the training and additional images are used for training teacher models. In other words, we use 604,388 samples as teachers' training samples. These samples are evenly distributed according to the number of teachers to ensure that the training samples of each teacher model do not overlap. A certain number of samples from the 26,032 test samples are reserved as the unlabeled public dataset, and the rest is used as the test samples of the student model. The MNIST dataset consists of handwritten digital pictures. There are 10 categories of pictures, corresponding to 10 Arabic numbers from '0' to '9' (LeCun et al., 1998). The numbers of training and testing image samples are 60,000 and 10,000, respectively. Each gray image sample has a size of $28 \times 28 \times 1$. Similar to SVHN, we use 60,000 samples as teachers' training samples, which are evenly distributed. A certain number of samples from the 10,000 test samples are reserved as the unlabeled public dataset, and the rest is used to test the student model accuracy. Our experiments use the same convolutional neural networks as in the prior work Papernot et al. (2018).

The cryptographic operations in SEDML only work with integers. However, the original voting counts will become floating-point numbers after Gaussian noises for differential privacy are added. Therefore, we need to convert floating-point numbers into decimal integers when doing secure comparison in ciphertext domain. In our SEDML, the maximum length of the number to be compared is 32 bits, and the maximum number of votes is 250 (given 250 teacher models), and the decimal bit corresponding to 32 bits is $10^9$. Therefore, for a floating-point number, we multiply it by $10^7$ and truncate it. Only the integer part is taken. Note that two numbers participating in the comparison will be multiplied with the same magnitude to retain the same expansion. Our SEDML protocol is implemented in Python. We run the experiments on a server with an AMD Ryzen 5 4600H CPU, 16GB RAM and the Windows 10 operating system.

### 5.2. Performance evaluation

We firstly stick with the SVHN dataset for comprehensive evaluations on both computation and communication performance.

#### 5.2.1. Computation performance

The computation cost of the secure aggregation procedure in SEDML consists of the following components in three phases. • **Phase 1: Secure Highest Vote Identification:** (line 5 to 11). In the first phase, SEDML needs to securely compare the voting results

**Table 1**
Computation Cost of Different Phases on the Cloud.

| Phase | Running time (s) |
|---|---|
| Highest Vote Identification | 23.249 |
| Threshold Check | 2.5119 |
| Consensus Label Identification | 21.575 |
| Overall | 47.3359 |

**Table 2**
Communication Cost of Different Phases on the Cloud.

| Phase | Cost (KB) |
|---|---|
| Highest Vote Identification | 29,628 |
| Threshold Check | 3292 |
| Consensus Label Identification | 28,201 |
| Overall | 61,121 |

for each pair of classes. We use the secure comparison algorithm to find the class with the highest vote. So we specifically evaluate the computation time of these operations. • **Phase 2: Secure Threshold Check:** (line 12 to 19). In the second phase, after the Gaussian noise with variance $\sigma_1^2$ is added to the highest vote, secure threshold check is utilized to determine whether the noisy maximum vote is greater than the pre-determined threshold. We evaluate the computation time for such secure threshold check. • **Phase 3: Secure Consensus Label Identification:** (line 20 to 32). Once the threshold check is passed, we enter the third phase of SEDML. To be precise, Gaussian noise with variance $\sigma_2^2$ is added to the votes of all classes, and then the maximum value is determined with the secure comparison algorithm among all the votes.

In our experiments, the number of samples used to query teacher models is 1000. In other words, the unlabeled dataset size held by the requester is 1000. To avoid variance in the result due to single round, we have performed 10 test runs and present the averaged cost. Table 1 reports the computation cost in each phase, for secure aggregation for 1000 samples (with 50 teachers). We can see that the running time of the secure highest vote identification phase is close to that of the secure consensus label identification phase. The latter is a bit smaller due to the fact that some samples do not pass the secure threshold check and they will not be involved in the third phase. As for the running time of the secure threshold check phase, it is only 2.5119 seconds, as only one secure comparison operation is needed. Note that the number of teachers has merely negligible impact on the computation cost at the server side. This is because the number of teachers is only related to the very first step of summing up the (encrypted) prediction vectors from the teachers, which has minimal cost. This can be seen from the fifth line of Algorithm 1, where it only needs to simply add up the teacher's votes in the secret sharing domain. In particular, for 50 teachers, the cost of such summation is as small as 0.01561 ms per sample. Therefore, the number of teachers have a negligible impact on the computation cost given in Table 1.

### 5.2.2. Communication performance

Regarding the communication performance, we examine and report the sizes of messages transmitted among the cloud servers, which are independent of computing platforms. The results are summarized in Table 2, with the number of teachers set to 50. It is noted that the number of teachers has no impact on the communication cost because secret-shared addition needs no interaction among the cloud servers. The size of messages transmitted in the secure consensus label identification is again slightly lower than that of secure highest vote identification phase. This is because some samples are filtered out in the intermediate secure threshold check phase. Overall, the data size communicated in the first and third phase is about 9 times of that in the second phase. Because for each sample, only one security comparison algorithm needs to be used in phase 2, while nine security comparisons are required in phase 1 and phase 3. Therefore, the sizes of messages transmitted in phase 1 and phase 3 is 9 times that in phase 2.

### 5.2.3. Scalability

We further investigate the scalability of the proposed SEDML from two aspects: the running time as a function of the number of samples, and the running time as a function of the number of classes. It should be noted that here we report the total running time of the whole secure aggregation procedure.

We first fix the number of classes to 10. The left subfigure in Fig. 4 details the computational time when the number of samples varies from 1000 to 5000. We can see that the running time scales linearly with the number of samples, ranging from 47.336s to 231.720s.

Then we fix the number of samples at 1000 while varying the number of classes. We set the number of classes as 10, 20, 30, 40, and 50, respectively. For this scalability experiment, we use synthetic data as we are measuring the running time whose complexity depends on the number of classes. The right subfigure in Fig. 4 shows the running time as the number of classes varies. We can see that the running time also increases *linearly* with the increase of the number of classes. This is because the total running time is dominated by the secure comparison operations. In the first and third phase, the complexity of required secure comparison operations is $O(n)$, while the second phase only needs a one-time secure comparison. For instance, when the number of classes is 10, 9 secure comparisons are required in the first phase; and when the number of classes is 20, 19 secure comparisons are required in the first phase. Therefore, the running time in SEDML as a function of the number of classes is also approximately $O(n)$ that is desirably linear.

## 5.3. Accuracy evaluation

We perform accuracy evaluation from the following aspects: • **Label Accuracy:** The ratio of the number of samples being correctly labelled through the secure aggregation process to the total number of samples. In calculating the label accuracy, we compare the aggregated label of each queried sample to its ground truth label, and count the number of correctly labelled samples whose aggregated labels match the ground truth labels. • **Student Model Accuracy:** It is simply the testing accuracy of the student model trained with samples that receive consensus labels through the secure aggregation process.

As described in Algorithm 1, a queried sample will be discarded if the teacher models cannot reach a consensus on the label through a threshold check. We set the threshold to 60% in the following experiments unless otherwise stated. Apparently, the label accuracy is directly related to the threshold setting. If a high threshold or a low threshold is used, the label accuracy is expected to decrease. We will give a detailed description later.

We compare the accuracy between SEDML and the plaintext baseline where the aggregation process is conducted on plaintext label prediction vectors. Following Xiang et al. (2020), we evaluate different settings of the $\epsilon$ parameter: 2.04, 5.035, and 8.03. For the SVHN and MNIST datasets, we set the $\delta$ to be $10^{-6}$ and $10^{-5}$ respectively. Because the scale of the SVHN we used is $10^6$ and the scale of the MNIST is $10^5$.

Figure 5 displays the evaluation results on the label accuracy and student model accuracy over the MNIST dataset. The accuracy evaluation results over the SVHN dataset are given in Fig. 6. Both the accuracy of MNIST and SVHN will decrease as the number of teachers increases. This is because the increase of the number of

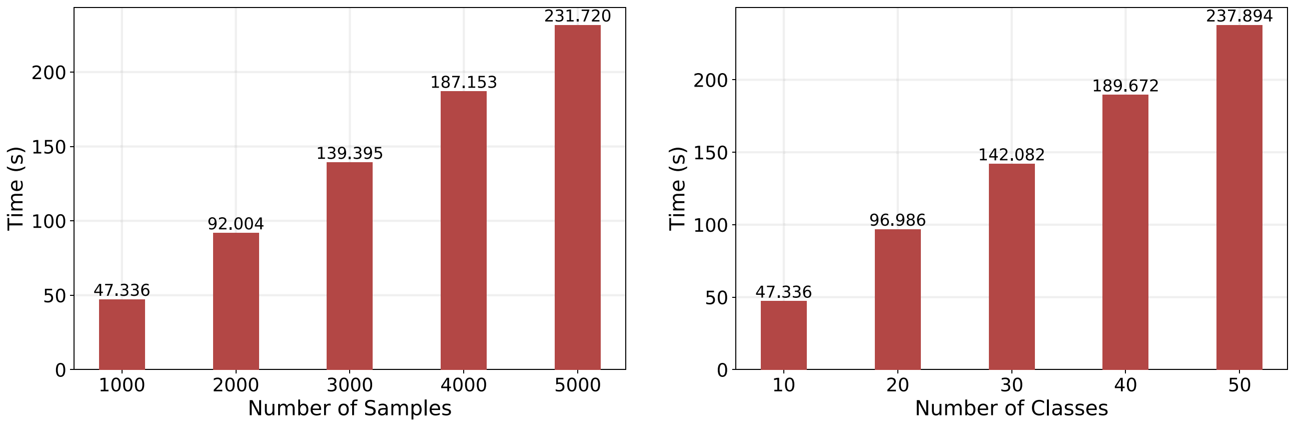

**Fig. 4.** The scalability of SEDML with varying number of samples (left) and varying number of classes (right).

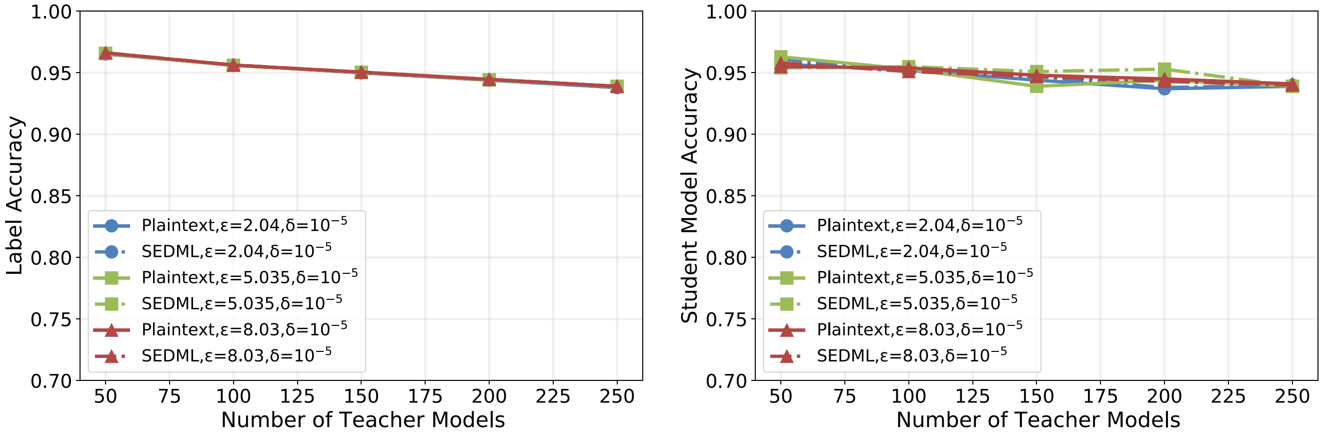

**Fig. 5.** Accuracy evaluation results on MNIST.

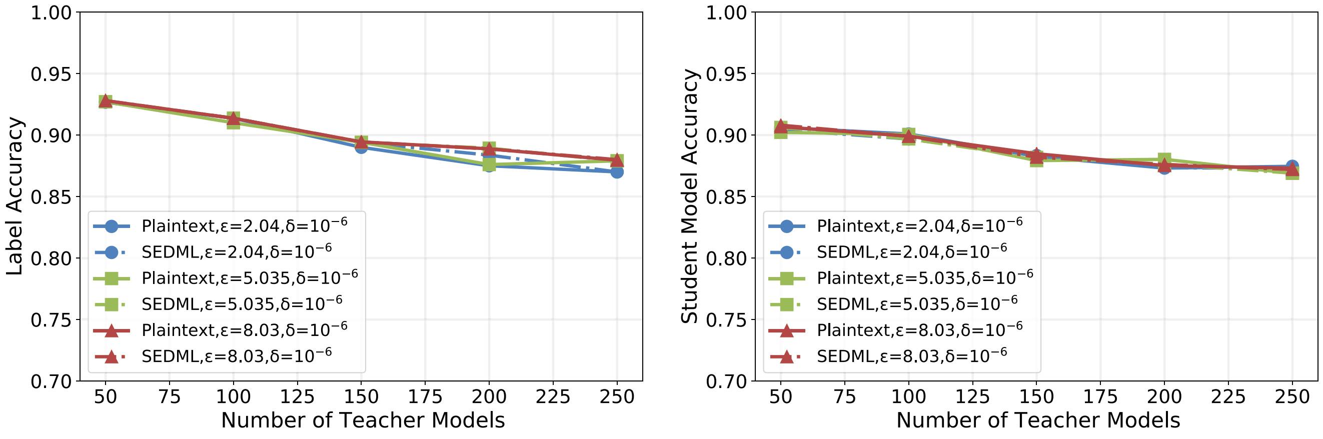

**Fig. 6.** Accuracy evaluation results on SVHN.

teachers will reduce the samples that each client can use for training the teacher model–the total number of training samples in our experiments is fixed. Therefore, the accuracy of the teacher model will be reduced, which in turn will also affect the accuracy of the student model.

It can be seen from the results that the accuracy gap between SEDML and the plaintext baseline is negligible. This demonstrates that SEDML promises efficiency in preserving privacy while retaining a comparable student model accuracy. This is because the secure comparison is only added in the ciphertext, and the other steps are consistent with the plaintext, so the accuracy will not be affected.

### 5.4. *Influence of the threshold*

We also investigate the influence of the threshold on the accuracy. The student model accuracy results for MNIST and SVHN are detailed in Fig. 7, when varying the threshold from 0.3 to 0.9. The number of queried samples is fixed to 9000 and the number of classes is fixed to 10. The left subfigure in Fig. 7 shows the student model accuracy on MNIST, where $\epsilon = 8.02$ and $\delta = 10^{-5}$. The right subfigure in Fig. 7 shows the student model accuracy on SVHN, where $\epsilon = 8.02$ and $\delta = 10^{-6}$. It is observed that the optimal threshold is between 0.5 and 0.6. Specifically, for the MNIST dataset, the optimal threshold is about 0.5, while for the

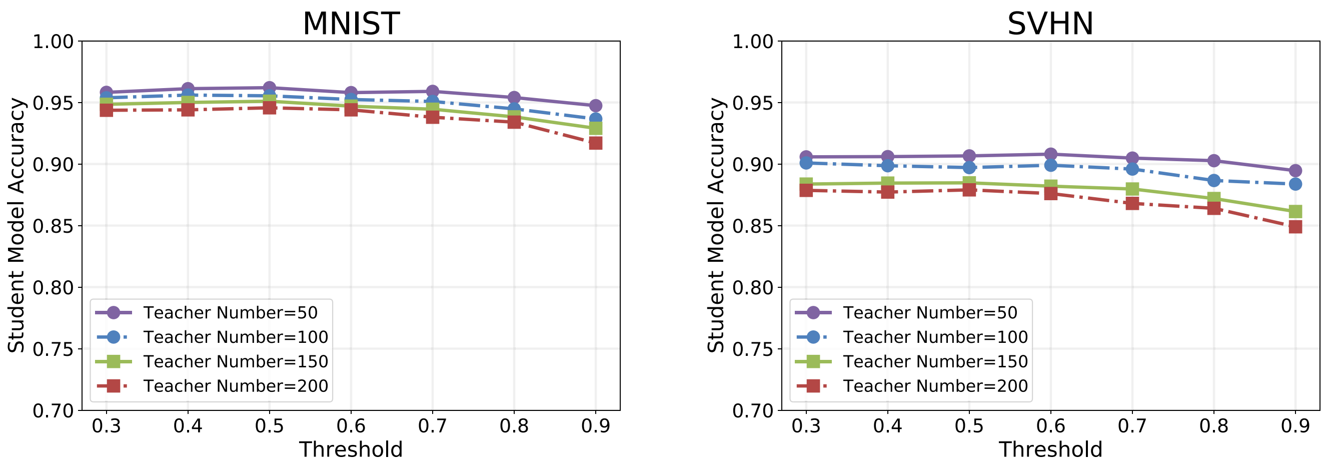

**Fig. 7.** The student model accuracy with varying thresholds and number of teacher models.

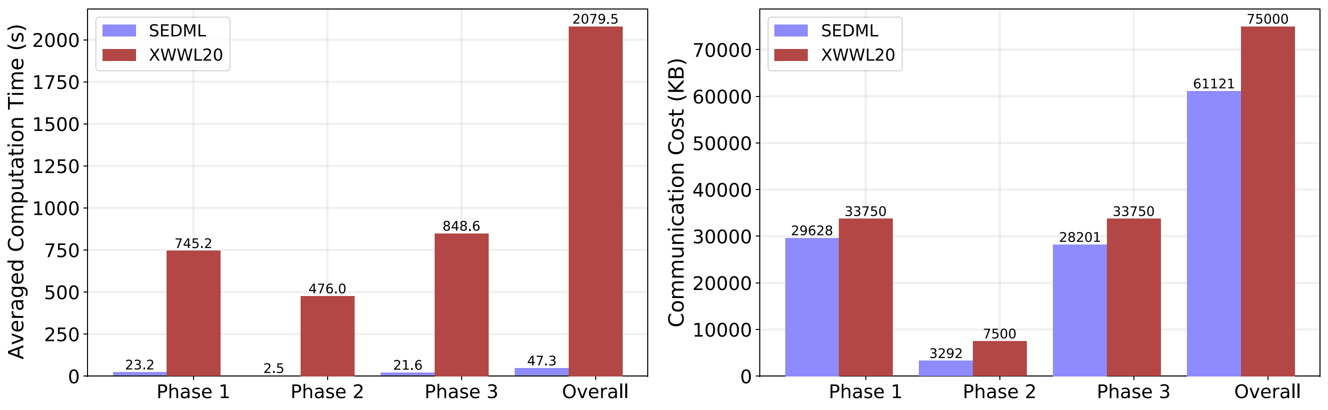

**Fig. 8.** Performance comparison with the prior work XWWL20 (Xiang et al., 2020) (over MNIST).

SVHN dataset, the optimal threshold is about 0.6. If the threshold is too small, some wrong labels with the same number of votes as the ground truth label will be falsely regarded as the final consensus label. This adds noisy training samples when training the student model, degrading its accuracy. If the threshold is too high, the votes of some correct labels can not exceed the threshold and are eventually discarded. In such case, less useful training samples are involved when training the student model, which can also deteriorate the student model accuracy. Therefore, the best threshold lies between 0.5 and 0.6. This empirical observation aligns with the prior work Xiang et al. (2020). This also accounts for why we choose 0.6 as the default threshold in our experiments.

### *5.5. Comparison with prior art*

We compare SEDML with the state-of-the-art privacy-preserving design by Xiang et al. (2020), simply referred to as XWWL20 in the following presentation. We use the MNIST dataset in the experiments for comparison. We first make comparison in the computation cost. The left subfigure in Fig. 8 compares the running time between SEDML and XWWL20. All phases in SEDML are at least 32× faster than XWWL20. This is because SEDML only builds on lightweight cryptography, as opposed to the expensive homomorphic encryption in XWWL20. The overall running time is 43× lower than that of XWWL20. We also compare the communication cost of SEDML with that of XWWL20. The results are given in the right subfigure of Fig. 8. Overall, our communication cost is 1.23× less than that of XWWL20. It is noted that Xiang et al. neither open source their implementation nor provide precise values for accuracy results matching our setup, so we have not been able to directly compare with their accuracy. We emphasize that we do not aim to achieve better accuracy than Xiang et al.'s work. Our focus instead is on securing the application of the PATE framework with superior efficiency.

## 6. Conclusion

To securely and efficiently harness the rich distributed isolated data, we have proposed SEDML, a new protocol that leverages the knowledge from distributed teacher models to train a student model. SEDML relies on the lightweight additive secret sharing to allow secure and efficient aggregation of the individual label predictions collected from the teacher models. We have performed extensive experimental evaluations on two popular real-world datasets MNIST and SVHN. The experiment results have demonstrated that the accuracy performance in SEDML is comparable to the plaintext baseline, and that SEDML greatly improves upon the Xiang et al. work in both computation and communication.

## Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## CRediT authorship contribution statement

**Yansong Gao:** Conceptualization, Methodology, Writing – original draft. **Qun Li:** Methodology, Formal analysis, Software.

**Yifeng Zheng:** Conceptualization, Supervision, Funding acquisition. **Guohong Wang:** Visualization, Data curation. **Jiannan Wei:** Validation, Writing – review & editing. **Mang Su:** Writing – review & editing, Validation.

## Acknowledgment

This work was supported in part by the National Natural Science Foundation of China (Grants 62002167 and 61702268), by the Natural Science Foundation of JiangSu Province (Grant BK20200461), by the Shenzhen Science and Technology Program (Grant RCBS20210609103056041), and by the Guangdong Basic and Applied Basic Research Foundation (Grant 2021A1515110027).

## References

Agrawal, N., Shamsabadi, A.S., Kusner, M.J., Gascón, A., 2019. QUOTIENT: two-party secure neural network training and prediction. In: Proc. of ACM CCS.
Alves, T.A., França, F.M., Kundu, S., 2019. MLPrivacyGuard: defeating confidence information based model inversion attacks on machine learning systems. In: Proc. of Great Lakes Symposium on VLSI, pp. 411–415.
Bakator, M., Radosav, D., 2018. Deep learning and medical diagnosis: a review of literature. Multimodal Technol. Interact. 2 (3), 47.
Beaver, D., 1991. Efficient multiparty protocols using circuit randomization. In: Proc. of CRYPTO.
Chen, X.W., Lin, X., 2014. Big data deep learning: challenges and perspectives. IEEE Access 2, 514–525.
China. China data security law. 2021. https://www.china-briefing.com/news/a-close-reading-of-chinas-data-security-law-in-effect-sept-1-2021/, accessed Sep 21, 2021.
Corrigan-Gibbs, H., Boneh, D., 2017. Prio: private, robust, and scalable computation of aggregate statistics. In: Poc. of USENIX NSDI, pp. 259–282.
Dwork, C., McSherry, F., Nissim, K., Smith, A.D., 2006. Calibrating noise to sensitivity in private data analysis. In: Proc. of TCC.
Erdogan E., Kupcu A., Cicek A.E.. UnSplit: data-oblivious model inversion, model stealing, and label inference attacks against split learning. 2021. arXiv preprint arXiv:2108.09033.
Europe. General data protection regulation. 2016. https://gdpr-info.eu/, accessed Feb 05, 2021.
Gao Y., Kim M., Thapa C., Abuadbba S., Zhang Z., Camtepe S.A., Kim H., Nepal S.. Evaluation and optimization of distributed machine learning techniques for internet of things. 2021. arXiv preprint arXiv:2103.02762.
Jordon, J., Yoon, J., van der Schaar, M., 2019a. Differentially private bagging: improved utility and cheaper privacy than subsample-and-aggregate. In: Wallach, H.M., Larochelle, H., Beygelzimer, A., d'Alché Buc, F., Fox, E.B., Garnett, R. (Eds.), Proc. of NeurIPS.
Jordon, J., Yoon, J., van der Schaar, M., 2019b. PATE-GAN: generating synthetic data with differential privacy guarantees. In: Proc. of ICLR.
Khosravy, M., Nakamura, K., Hirose, Y., Nitta, N., Babaguchi, N., 2021. Model inversion attack: analysis under gray-box scenario on deep learning based face recognition system. KSII Trans. Internet Inf.Syst. (TIIS) 15 (3), 1100–1118.
Knott, B., Venkataraman, S., Hannun, A., Sengupta, S., Ibrahim, M., van der Maaten, L., 2020. CRYPTEN: secure multi-party computation meets machine learning. In: Proc. of NeurIPS Workshop on Privacy-Preserving Machine Learning.
LeCun, Y., Bengio, Y., Hinton, G., 2015. Deep learning. Nature 521 (7553), 436.
LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., 1998. Gradient-based learning applied to document recognition. Proc. IEEE 86 (11), 2278–2324.
Li Z., Zhang Y.. Label-leaks: membership inference attack with label. 2020. arxiv e-prints ArXiv–2007.
Liu, X., Zheng, Y., Yuan, X., Yi, X., 2021. MediSC: towards secure and lightweight deep learning as a medical diagnostic service. In: Proc. of ESORICS.
Long, Y., Wang, B., Yang, Z., Kailkhura, B., Zhang, A., 2021. G-PATE: scalable differentially private data generator via private aggregation of teacher discriminators. In: Proc. of NeurIPS.
McMahan, B., Moore, E., Ramage, D., Hampson, S., Arcas, B.A.y., 2017. Communication-efficient learning of deep networks from decentralized data. In: Proc. of AISTATS.
Mironov, I., 2017. Rényi differential privacy. Prof. of IEEE CSF.
Mishra, P., Lehmkuhl, R., Srinivasan, A., Zheng, W., Popa, R.A., 2020. Delphi: a cryptographic inference service for neural networks. In: Proc. of USENIX Security.
Mohassel, P., Zhang, Y., 2017. SecureML: a system for scalable privacy-preserving machine learning. In: Proc. of SP. IEEE, pp. 19–38.
Najafabadi, M.M., Villanustre, F., Khoshgoftaar, T.M., Seliya, N., Wald, R., Muharemagic, E., 2015. Deep learning applications and challenges in big data analytics. J. Big Data 2 (1), 1–21.
Netzer Y., Wang T., Coates A., Bissacco A., Wu B., Ng A.Y.. Reading digits in natural images with unsupervised feature learning. 2011.
Papernot, N., Abadi, M., Erlingsson, U., Goodfellow, I., Talwar, K., 2017. Semi-supervised knowledge transfer for deep learning from private training data. In: Proc. of ICLR.
Papernot, N., Song, S., Mironov, I., Raghunathan, A., Talwar, K., Erlingsson, U., 2018. Scalable private learning with PATE. In: Proc. of ICLR.
Privacy C.. Tf encrypted: Encrypted deep learning in tensorflow. 2020. Online at https://tf-encrypted.io/.
Rahman, M.A., Rahman, T., Laganière, R., Mohammed, N., Wang, Y., 2018. Membership inference attack against differentially private deep learning model. Trans. Data Priv. 11 (1), 61–79.
Riazi, M.S., Samragh, M., Chen, H., Laine, K., Lauter, K., Koushanfar, F., 2019. XONN: XNOR-based oblivious deep neural network inference. In: Proc. of USENIX Security, pp. 1501–1518.
Riazi, M.S., Weinert, C., Tkachenko, O., Songhori, E.M., Schneider, T., Koushanfar, F., 2018. Chameleon: a hybrid secure computation framework for machine learning applications. In: Proc. of ACM AsiaCCS.
States U.. California privacy rights act. 2020. https://www.cookiebot.com/en/cpra/, accessed Feb 05, 2021.
Wang, B., Wu, F., Long, Y., Rimanic, L., Zhang, C., Li, B., 2021. DataLens: scalable privacy preserving training via gradient compression and aggregation. In: Proc. of ACM CCS.
Wang, Q., Gao, J., Lin, W., Yuan, Y., 2019. Learning from synthetic data for crowd counting in the wild. In: Proc. of CVPR, pp. 8198–8207.
Xiang, L., Wang, L., Wang, S., Li, B., 2020. Achieving consensus in privacy-preserving decentralized learning. In: Proc. of IEEE ICDCS.
Zheng, Y., Duan, H., Wang, C., 2019. Towards secure and efficient outsourcing of machine learning classification. In: Proc. of ESORICS.

**Yansong Gao** received his Bachelor degree from Henan University of Science and Technology, M.Sc degree from the University of Electronic Science and Technology of China in 2013 and a PhD degree from the University of Adelaide, Australia, in 2017. He was with Data61, CSIRO, Sydney, Australia as a Postdoc Research Fellow. He is with Nanjing University of Science and Technology, China, as an associate professor. His current research interests are AI security and privacy, hardware security, and system security.

**Qun Li** is currently pursing his Master degree at Nanjing University of Science and Technology. He received Bachelor degree from the Qingdao university of Science and Technology in 2020. His research interest is in privacy protection.

**Yifeng Zheng** is an assistant professor with the School of Computer Science and Technology, Harbin Institute of Technology, Shenzhen, China. He received the BEng degree in information engineering from the South China University of Technology, Guangzhou, China, in 2013, and the PhD degree in computer science from the City University of Hong Kong, Hong Kong, China, in 2019. He worked as a postdoc in the Commonwealth Scientific and Industrial Organization (CSIRO), Australia, and the City University of Hong Kong. His current research interests are focused on security and privacy related to cloud computing, IoT, machine learning, and multimedia.

**Guohong Wang** is currently pursing his PhD degree at the Nanjing University of Science and Technology. He received her Bachelor degree from the Xi'an Jiaotong University City College. His main research interests including AI security and privacy.

**Jiannan Wei** received the MS degree from Zhengzhou University, China, in 2012, and the PhD degree from the School of Computer Science and Software Engineering, University of Wollongong, Australia. She currently is Assistant Professor with the School of Computer Science and Engineering, Nanjing University of Science and Technology, China. Her major research interests include public key cryptography, vehicular crowdsensing network, privacy-preserving digital signatures, and wireless network security.

**Mang Su** received her PhD degree from Xidian University, China, in 2014. Currently, she is an associate professor at the school of computer science and engineering at Nanjing University of Science and Technology, China. She has authored or coauthored over 30 technical papers, including being published in international journals and conferences, such as the Information Sciences, IEEE ICC, IEEE TrustCom and IEEE GLOBECOM. Her research interests include cloud security and privacy protection.