Privacy-Preserving Machine Learning: Methods, Challenges and Directions

Runhua Xu[1]*, **Nathalie Baracaldo**[1], **and James Joshi**[2†]

[1] IBM Research - Almaden Research Center, San Jose, CA, United States, 95120
[2] School of Computing and Information, University of Pittsburgh, Pittsburgh, PA, United States, 15260
runhua@ibm.com, baracald@us.ibm.com, jjoshi@pitt.edu

Abstract

Machine learning (ML) is increasingly being adopted in a wide variety of application domains. Usually, a well-performing ML model relies on a large volume of training data and high-powered computational resources. Such a need for and the use of huge volumes of data raise serious privacy concerns because of the potential risks of leakage of highly privacy-sensitive information; further, the evolving regulatory environments that increasingly restrict access to and use of privacy-sensitive data add significant challenges to fully benefiting from the power of ML for data-driven applications. A trained ML model may also be vulnerable to adversarial attacks such as membership, attribute, or property inference attacks and model inversion attacks. Hence, well-designed privacy-preserving ML (PPML) solutions are critically needed for many emerging applications. Increasingly, significant research efforts from both academia and industry can be seen in PPML areas that aim toward integrating privacy-preserving techniques into ML pipeline or specific algorithms, or designing various PPML architectures. In particular, existing PPML research cross-cut ML, systems and applications design, as well as security and privacy areas; hence, there is a critical need to understand state-of-the-art research, related challenges and a research roadmap for future research in PPML area. In this paper, we systematically review and summarize existing privacy-preserving approaches and propose a ***P**hase*, ***G**uarantee*, and ***U**tility* (***PGU***) triad based model to understand and guide the evaluation of various PPML solutions by decomposing their privacy-preserving functionalities. We discuss the unique characteristics and challenges of PPML and outline possible research directions that leverage as well as benefit multiple research communities such as ML, distributed systems, security and privacy.

Key Phrases: Machine Learning, Privacy-Preserving Machine Learning

*Part of this work was done while Runhua Xu was at the School of Computing and Information, University of Pittsburgh.

†This work was performed while James Joshi was serving as a program director at NSF; and the work represents the views of the authors and not that of NSF.

Preprint. We are updating it to further refine the paper.

Contents

1 Introduction

Machine learning (ML) is increasingly being applied in a wide variety of application domains. For instance, emerging deep neural networks, also known as deep learning (DL), have shown significant improvements in model accuracy and performance, especially in application areas such as computer vision, natural language processing, and speech or audio recognition [1, 2, 3]. Emerging federated learning (FL) is another collaborative ML technique that enables training a high-quality model while training data remains distributed over multiple decentralized devices [4, 5]. FL has shown its promise in various application domains, including healthcare, vehicular networks, intelligent manufacturing, among others [6, 7, 8]. Although these models have shown considerable success in AI-powered or ML-driven applications, they still face several challenges, such as (i) lack of powerful computational resources and (ii) availability of huge volumes of data for model training. In general, the performance of an ML system relies on a large volume of training data and high-powered computational resources to support both the training and inference phases.

To address the need for computing resources with high-performance CPUs and GPUs, large memory storage, etc., existing commercial ML-related infrastructure service providers, such as Amazon, Microsoft, Google, and IBM, have devoted significant amounts of their efforts toward building *infrastructure as a service* (IaaS) or *machine learning as a service* (MLaaS) platforms with appropriate rental fees. The resource-limited clients can employ ML-related IaaS or MLaaS to manage and train their models first and then provide data analytics and prediction services through their applications directly.

Availability of massive volumes of training data is another challenge for ML systems. Intuitively, more training data indicates better performance of an ML model; thus, there is a need for collecting large volumes of data and in many cases from multiple sources. However, the collection and use of the data, as well as the creation and use of ML models, raise serious privacy concerns because of the risks of leakage of private or confidential information. For instance, recent data breaches have significantly increased the privacy concerns of large-scale collection and use of personal data [9, 10]. An adversary can also infer private information by exploiting an ML model via various inference attacks such as membership inference attacks [11, 12, 13, 14, 15], model inversion attacks [16, 17, 18], property inference attacks [19, 20], and privacy leakage from gradients exchanged in distributed ML scenarios [21, 22]. For instance, in an membership inference attack, an attacker can infer whether or not data related to a particular patient has been included in the training of an HIV-related ML model. In addition, existing regulations such as the Health Insurance Portability and Accountability Act (HIPPA) and more recent ones such as the European General Data Protection Regulation (GDPR), Cybersecurity Law of China, California Consumer Privacy Act (CCPA), etc., increasingly restrict the availability and use of privacy-sensitive data. Such privacy concerns and challenges pose significant roadblocks to the adoption of ML models for real-world applications.

To tackle the increasing privacy concerns related to using ML in applications, in which users' privacy-sensitive data such as electronic health/medical records, location information, etc., are stored and processed, it is crucial to devise innovative privacy-preserving ML (PPML) solutions. More recently, there have been increasing efforts focused on PPML research that integrate existing anonymization mechanisms into ML pipelines or design innovative new privacy-preserving methods and architectures for ML systems. Recent surveys focused on ML, including Federated Learning such as in [23, 24, 25, 26, 27, 28] partially illustrate or discuss the specific privacy and security issues in ML or FL systems. Each existing PPML approach addresses part of privacy concerns or is only applicable to limited scenarios. There is no unified or or holistic view of PPML solutions. For instance, the adoption of differential privacy in ML systems can lead to model utility loss, e.g., reduced model accuracy. Similarly, the use of secure multi-party computation approaches incurs high communication overhead or computation overhead. The communication overhead is caused by transmitting a large volume of intermediate data, such as garbled tables of circuit gates. At the same time, the adoption of advanced cryptosystems [29, 30] leads to computation overhead.

ML security, such as issues of stealing the ML models, injecting Trojans and availability of ML services and corresponding countermeasures, have been discussed in various recent articles such as those related to the systematization of knowledge [31], or surveys/analyses [32, 33, 34]. However, there is still a lack of systematization of knowledge discussion and evaluation with privacy as the key focus. Inspired by the CIA (confidentiality, integrity and availability) triad in security designed to more holistically understand information security, in this paper, we propose a ***PGU*** triad

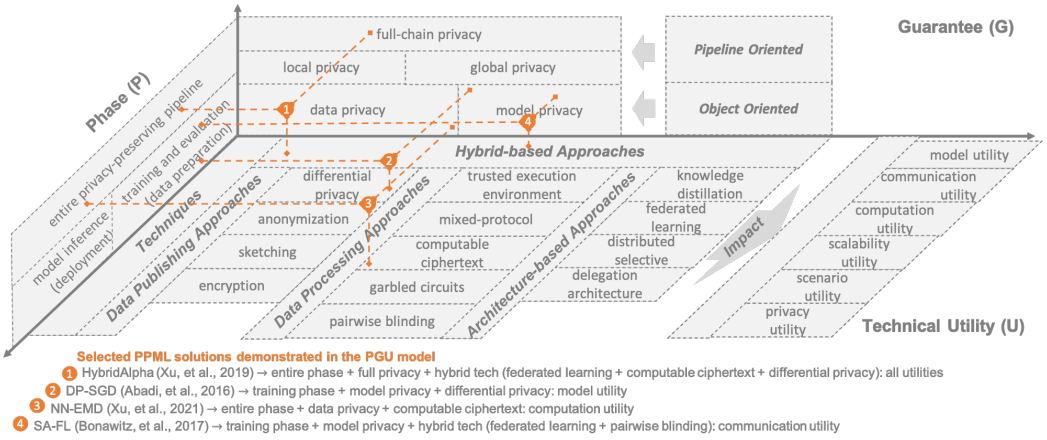

Figure 1: An overview of PGU model to evaluate the privacy-preserving machine learning systems and illustration of selected PPML examples in the PGU model. The demonstrated PPML examples in the figure are HybridAlpha [35], DP-SGD [36], NN-EMD[37], SA-FL[38].

- referring to *Phase*, *Guarantee*, and *technical Utility* - to better comprehend and help guide the understanding or evaluation of various PPML solution space by decomposing functionalities/features of privacy-preserving approaches, as illustrated in Figure 1. Here, *phase* represents the phase of privacy-preserving functionalities that occur in various phases of a ML pipeline; *guarantee* denotes the strength or scope of the privacy protection under a given set of threat models and/or trust assumptions; *utility* captures the impact of adopted privacy solutions on the accuracy or usefulness of computational results of ML systems. Based on the PGU analysis framework, we also discusses various challenges and potential future research directions in PPML.

More specifically, we first introduce the general ML pipeline in a nutshell. Then we discuss the PPML pipeline from various phases of privacy-preserving functionalities that occur in the process-chain in these systems; these include *privacy-preserving data preparation*, *privacy-preserving model training and evaluation*, *privacy-preserving model deployment*, and *privacy-preserving model inference*.

Following that, we discuss the privacy guarantees provided by existing PPML solutions by analyzing the strength and/or scope of the privacy protection from two perspectives: object-oriented privacy protection and pipeline-oriented privacy protection, based on common threat model settings and trust assumptions. From an object-oriented perspective, PPML solutions either aim to protect *input privacy* by preventing the leakage/exposure of private information from training or inference data samples, or to protect *model privacy* by mitigating privacy disclosure from the learned model. From a pipeline viewpoint, PPML solutions are concerned with the privacy-preserving functionality associated with an entity or collection of entities in the pipeline of a machine learning solution. Typically, it encompasses local privacy, global privacy, and full-chain privacy, all of which will be described in further detail later.

Additionally, we investigate the technical utility of various PPML systems. We begin by dissecting and classifying existing PPML solutions into four categories: *data publishing* approaches, *data processing* approaches, *architecture based* approaches and *hybrid* approaches. Then, we examine their impact on an ML system's utility, including *computation utility*, *communication utility*, *model utility*, *scalability utility*, *scenario utility*, among others.

Finally, we discuss the challenges of designing PPML solutions and future research directions.

Organization. The remainder of this paper is organized as follows. We briefly present the ML pipeline in Section 2 by reviewing the critical tasks in ML-related systems, and third-party facility-related ML solutions. In Section 3, we present general discussion of existing privacy-preserving methods by considering the phases where they are applied, and discuss privacy guarantees in Section 4. We investigate technical utility of PPML solutions in Section 5 by summarizing and classifying privacy-preserving techniques and their impact on ML systems. Furthermore, we also discuss the challenges and open problems in PPML solutions and outline the promising directions of future research in Section 6. Finally, we conclude the paper in Section 7.

2 Machine Learning Pipeline in a Nutshell

The four phases of a machine learning system are typically as follows: *data preparation or preprocessing*, *model training and evaluation*, *model deployment*, and *model inference*. More broadly, the ML pipeline can be divided into *training* and *serving* phases. In this scenario, model *training* encompasses processes such as data collection and preprocessing, model training, and model evaluation; while model *serving* mostly focuses on how to use a trained model, such as how to deploy the model and infer the result given a certain data sample.

In this section, we first formally differentiate computational tasks between the model training and model serving, and then based on this, we discuss privacy-preserving training and privacy-preserving serving approaches. Following that, we demonstrate a general machine learning pipeline that utilizes both self-owned and third-party infrastructure to handle the majority of machine learning-based workloads. The processing pipeline is divided into three trust domains: *trusted data owner*, *trusted third-party*, and *trusted model user*. On this basis, we may examine the privacy guarantees provided by existing PPML solutions by considering various trust assumptions and probable adversarial threats.

2.1 Computation Tasks in Model Training and Serving

From the perspective of underlying computation tasks, there is no strict boundary between the model training and the model serving (i.e., inference) phases. The computed function in the serving procedure could be viewed as a simplified version of the training procedure without loss computation, regularization, (partial) derivatives, and model weights update, which are needed during training phase. For instance, in a stochastic gradient descent (SGD) based training approach, the computation that occurs at the inference phase could be viewed as one round of computation in the training phase without operations related to model gradients update. In a more complex neural networks, the computation involved during the training phase includes continuously feeding a set of data to the designed network for multiple training epochs. In contrast, the inference service can be treated as only one epoch of computation for one data sample to predict a label without a propagation procedure and related a regularization or normalization step.

Formally, given a set of training samples denoted as $(\boldsymbol{x}_1, y_1)$, ..., $(\boldsymbol{x}_n, y_n)$, where $\boldsymbol{x}_i \in \mathbb{R}^m$, $y_i \in \mathbb{R}$, the goal of a ML model training (for simplicity, assume a linear model) is to learn a fit function denoted as

$$f_{\boldsymbol{w},b}(\boldsymbol{x}) = \boldsymbol{w}^\intercal \boldsymbol{x} + b, \tag{1}$$

where $\boldsymbol{w} \in \mathbb{R}^m$ is the set of model parameters, and b is the intercept. To find proper model parameters, usually, we need to minimize the regularized training error given by

$$E(\boldsymbol{w}, b) = \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(y_i, f(\boldsymbol{x}_i)) + \alpha R(\boldsymbol{w}), \tag{2}$$

where $\mathcal{L}(\cdot)$ is a loss function that measures model fit and $R(\cdot)$ is a regularization term (a.k.a., penalty) that penalizes model complexity; α is a non-negative hyperparameter that controls the regularization strength. Regardless of various choices of $\mathcal{L}(\cdot)$ and $R(\cdot)$, stochastic gradient descent (SGD) is a common optimization method for unconstrained optimization problems. A simple SGD method *iterates* over the training samples and for each sample updates the model parameters according to the update rule given by

$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta \nabla_{\boldsymbol{w}} E = \boldsymbol{w} - \eta [\alpha \nabla_{\boldsymbol{w}} R + \nabla_{\boldsymbol{w}} \mathcal{L}] \tag{3}$$

$$b \leftarrow b - \eta \nabla_b E = b - \eta [\alpha \nabla_b R + \nabla_b \mathcal{L}] \tag{4}$$

where η is the learning rate which controls the step-size in the parameter space.

Given the trained model $(\boldsymbol{w}_{\text{trained}}, b_{\text{trained}})$, the goal of the model serving is to predict a value $\hat{y}$ for target sample $\boldsymbol{x}$ as follows:

$$\hat{y} = f_{\boldsymbol{w}_{\text{trained}}, b_{\text{trained}}}(\boldsymbol{x}). \tag{5}$$

As illustrated above, the computed functions in the inference phase (i.e., Equation (5)), is part of computed procedures in the SGD training (i.e., Equation (2)). Similarly, in the case of a deep neural network, the model training process is a model inference process with extra back-propagation to compute the partial derivative of weights. This also indicates that the task of privacy-preserving

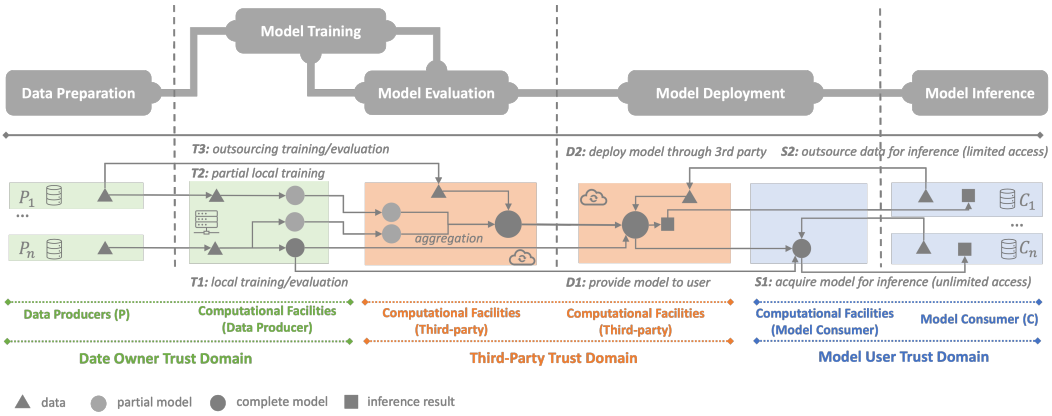

Figure 2: An illustration of machine learning pipeline (above part) and demonstration of corresponding processes showing different trust domains (bottom part) in various scenarios of ML applications.

training is more challenging than the task of privacy-preserving serving. While the majority of extant privacy-preserving training solutions that rely on secure computation approaches imply the possibility of achieving privacy-preserving serving even when the proposal does not explicitly declare it; this is not true for vice versa. A more specific demonstration is presented in Section 5.2.

2.2 An Illustration of Trusted Third-party based ML Pipeline

As seen in figurename 2, we provide a high-level overview of the machine learning pipeline, its associated process chain, and relying facilities, which include the data owner's own devices and third-party-provided resources (e.g., IaaS, PaaS, and MLaaS). Note that third-party facility-related ML pipeline is also a widespread adoption in recent ML-related applications. The pipeline is divided into four stages: *data preparation* stage where data is collected and preprocessed; *model training and evaluation* stage where an ML algorithm is used to train an ML model and the trained model is evaluate; textitmodel deployment stage that involves steps to provision the model to the target user or deploy it through a third-party (e.g., as a service); and *model serving* stage where the model is used by the user to obtain the prediction/inference results.

From the perspective of the types of participants, the ML pipeline includes three entities/roles: the *data producer (DP)*, the *model consumer (MC)*, and the *computational facility (CF)*; the CF may be owned by the data producer or model consumer themselves, or is employed by a trusted third-party. The *data producer* owns and provides the training data to train various ML models. Simultaneously, the *model consumer* possesses massive amounts of target data and expects acquiring various machine learning inference services such as labeling the target data, predicting values, and clustering the target data into groups.

For model training/evaluation or model deployment/serving stages, there exist two possible cases: the data producer (or the model consumer) (i) intends to use locally owned CFs, or (ii) prefers to employ third-party CFs instead of local CFs. As a result, different computation options exist for model training/evaluation and model deployment/serving. As illustrated in Figure 2, in case (i), the *data producer* can train a complete model locally (T1) or locally train a partial model that can be used to synthesize a global ML model in a collaborative or distributed manner (T2). In case (ii), the *data producer* directly sends out its data to third-party entities that have computational facilities to employ their computational resources to train an ML model (T3). Such third-party facilities may include the *edge nodes* in an edge computing environment and/or IaaS servers in an cloud computing environment. Accordingly, the *model consumer* may also acquire the trained model directly (D1) for model inference service with unlimited access (S1) if it has local computational facilities; otherwise, the *model consumer* can also utilize a third-party facility where the trained model is deployed (D2) to acquire the prediction service (S2).

From the perspective of privacy-preserving phases in an ML pipeline, we can classify PPML research into two directions: privacy-preserving training phase including private data preparation and model

generation and the privacy-preserving serving phase involving private model deployment and serving. We discuss the affected phases of privacy-preserving approaches in Section 3 in detail.

It is also important to consider the trust domains to characterize the trust assumptions and related threat models. We divide the ML system into three domains: the data owner's local trust domain, the third-party CF trust domain, and the model user's trust domain. Based on those trust domains, we analyze various types of privacy guarantees provided by a PPML system. We present more details in Section 4.

From the perspective of underlying techniques, we decompose recently proposed PPML solutions to summarize and classify the critical privacy-preserving components to evaluate their potential utility impact. Intuitively, the underlying privacy-preserving techniques such as differential privacy, conventional multi-party computation building on the garbled circuits and oblivious transfer, or customized secure protocols are widely employed in the PPML solutions. Besides, various well-designed learning architectures have been broadly studied under specific trust domains and threat models. Furthermore, emerging advanced cryptosystems such as homomorphic encryption and functional encryption also show their promise for PPML with strong privacy guarantees. More detailed taxonomy and analysis will be introduced in Section 5.

3 Privacy-Preserving Phases in PPML

In this section we present existing PPML solutions considering the privacy-preserving phases in an ML pipeline. Figure 2 illustrates four phases in a typical ML pipeline: *data preparation*, *model training and evaluation*, *model deployment* and *model serving*. Correspondingly, the existing PPML pipeline involves (i) *privacy-preserving data preparation*, (ii) *privacy-preserving model training and evaluation*, (iii) *privacy-preserving model deployment*, (iv) *privacy-preserving inference*. For simplicity, we analyze the PPML mainly by focusing on *privacy-preserving model generation* covering phases (i-ii) and *privacy-preserving model serving* including phases (iii-iv).

3.1 Privacy-Preserving Model Generation

Most privacy-preserving model generation solutions emphasize that the adopted privacy-preserving approaches should prevent the leakage of private information in the training data from leaking the trusted scope of the data sources. In particular, the key privacy leakage issues to consider during model generation relate to *data* and *computation*; existing research address these through follow two key research questions:

(i) how to distill/filter the training data so as to minimize or completely remove any privacy sensitive information;

(ii) how to computationally process the training data in a privacy preserving manner.

3.1.1 Privacy-Preserving Data Preparation

From the perspective of ***data***, existing privacy-preserving approaches focus on the following directions: (i) adopting the traditional anonymization mechanisms such as k-anonymity[39], l-diversity[40], and t-closeness [41] to remove the identifier information in the training data before using the data for training; (ii) representing the raw dataset using a surrogate dataset by grouping the anonymized data [42] or abstracting the dataset by sketch techniques [43, 44]; (iii) employing differential privacy mechanisms [45, 46, 47] to add privacy budget (noise) into the dataset to avoid private information leakage.

Specifically, in [48], Friedman et al. try to providing k-anonymity in the data mining algorithm, while the works in [49, 50] focus on the utility metric and provide a suite of anonymization algorithms to produce an anonymous view based on ML workloads. Besides, recently, differential privacy mechanism has shown its promise in emerging DL models that rely on training on large datasets. For example, Abadi et al. [36] proposes a differentially private stochastic gradient descent approach to train a privacy-preserving DL model. Among more recent work, McMahan et al. [51] demonstrates that it is possible to train large recurrent language models with user-level differential privacy guarantees with only a negligible cost in predictive accuracy. Recent parameter-transfer meta-learning (i.e., the applications including few-shot learning, federated learning, and reinforcement learning) often

requires the task-owners to share model parameters that may result in privacy disclosure. Proposals of privacy-preserving meta-learning such as those proposed by Xu et al. [35] and Geyer et al. [52] address the problem of private information leakage in federated learning (FL) by proposing an algorithm to achieve client-sided (local) differential privacy. In [53], Li et al. formalize the notion of task-global differential privacy and proposes a differentially private algorithm for gradient-based parameter transfer that satisfies the privacy requirement as well as retains provable transfer learning guarantees in convex settings.

Thanks to recent successful work related to computing over encrypted data (i.e., practical computation over the encrypted data), training ML models on encrypted data is emerging as a promising approach to protecting privacy of training data. Unlike the traditional anonymization mechanisms or differential privacy mechanisms that are still susceptible to the inference or de-anonymization attacks, such as demonstrated in [54, 55, 11, 56], wherein an adversary may have additional background knowledge, the encryption based approaches can provide a stronger privacy guarantees - called *confidential-level privacy* in the rest of the paper. Hence these encryption based approaches are receiving more and more attention in recently [57, 58, 35, 59, 60, 61, 62, 63, 64], wherein the training data or the transferred model parameter is protected by cryptosystems while still allowing the subsequent computation outside of the trusted scope of the data sources.

3.1.2 Privacy-Preserving Model Training

From a ***computation*** standpoint, existing privacy-preserving approaches are also correspondingly divided into two directions: (i) for the case that the training data is processed by employing conventional anonymization or differential privacy mechanisms, the computation involved during training is as is done in a vanilla model training; (ii) for the case that the training data is protected via cryptosystems, due to the confidential-level privacy, the computation involved in privacy-preserving (i.e., crypto-based) training is a bit more complex than normal model training. The demand of training computation over the ciphertext indicates that the direct use of traditional cryptosystems such as AES and DES is not applicable here, as those cryptosystems only secure data rather than operating the ciphertext. That crypto-based training makes use of recently proposed advanced cryptographic schemes that primarily include homomorphic encryption [29, 65, 66, 67, 68] and functional encryption[30, 69, 70, 71, 72, 73, 74] schemes; these enable computation over the encrypted data. In general, homomorphic encryption (HE) is a form of public encryption that enables computation over encrypted data without requiring the data to be decrypted. In HE, the result of the computation remains encrypted and is the encrypted version of the result that would be obtained when the same computation is conducted on the original data. Similarly, functional encryption (FE) is a generalization of public-key encryption in which the holder of a functional decryption key is able to learn a function of what the ciphertext is encrypting without learning the protected inputs themselves.

Note that. compared to the non-crypted approaches, training over encrypted data may involve an additional step - *data conversion* or *data encoding*. The reason for this step is that the majority of those cryptosystems, such as multi-party functional encryption [70, 75] and BGV scheme [76] (i.e., an implementation homomorphic encryption), are built on the integer group, whereas the training data preprocessed utilizing widely used methods such as feature encoding, discretization, normalization, or the model parameter exchanged is always in floating-point numbers. It is worth noting that this is not a requirement for all crypto-based training methods. For instance, an emerging implemented CKKS scheme [77] - an instance of homomorphic encryption - can support approximate arithmetic computation.

Typically, data conversion consists of two operations: encoding and decoding. The encoding phase is typically used to transform floating-point values to integers, which enables the data to be encrypted and then used in cryptographic-based training. On the contrary, the decoding step is used to recover the floating-point numbers from the trained model or crypto-based training result. Without a doubt, the accuracy and efficiency of those rescaling processes are dependent on the conversion precision level. In Section 5, we will discuss the potential impact of data conversion in further detail.

3.2 Privacy-Preserving Model Serving

There is no apparent distinction between privacy-preserving model deployment and model inference in the majority of PPML systems; so, we refer to the discussion in this section as privacy-preserving model serving.

Compared to privacy-preserving training, tackling privacy-preserving model serving challenges is relatively simpler from a computational standpoint, as demonstrated in Section 2.1. Except for the emerging machine learning models of complex deep neural networks, there are few studies devoted exclusively to privacy-preserving inference. We observe that the majority of PPML solutions that make use of advanced cryptosystems (primarily homomorphic encryption and related schemes) are limited to privacy-preserving inference, as these crypto-based solutions are inefficient when applied to the complex and massive training computations of neural networks. We also note that these solutions primarily secure inference data samples, trained models, or both.

Additionally, another subfield of privacy-preserving model serving research focuses on privacy-preserving model querying or publication in cases when the trained model is deployed in a privacy-preserving manner, with the model consumer and model owner typically separated. The primary question here is how to prevent an adversarial model user from inferring the private information from the original training data. According to various model inference attack assumptions, an adversary has limited (or unlimited) access times to query the trained model. Furthermore, the adversary possesses (or lacks) additional knowledge about the trained model specification, which is referred to as white-box (or black-box) attacks. To address those inference attacks, a naive privacy-preserving strategy would be to restrict number of queries or to decrease the background information related to the disclosed model for a specific model user. Beyond that, potential preventative approaches include the following:

(i) *private aggregation of teacher ensembles (PATE)* approaches [78, 79, 80], wherein the knowledge of an ensemble of "teacher" models is transferred to a "student" model, with intuitive privacy provided by training teachers on disjoint data, and strong privacy ensured by noisy aggregation of teachers' responses;

(ii) *model transformation* approach such as MiniONN [81] and variant solutions as in [82], where an existing model is transformed into an oblivious neural network supporting privacy-preserving predictions with reasonable efficiency;

(iii) *model compression* approach, especially applied in the emerging deep learning model with a large set of model parameters, where knowledge distillation methods [83, 84] are adopted to compress the deep neural networks model. While knowledge distillation's primary objective is to minimize the size of the deep neural network model, it also provides extra privacy-preserving capabilities [85, 86]. Intuitively, the distillation technique eliminates redundant information in the model and decreases the probability that the attacker may infer potential private information in the model via repetitive queries.

3.3 Full Privacy-Preserving Pipeline

The notion of a *full privacy-preserving pipeline* is rarely mentioned in PPML proposals. Existing PPML solutions either enable *privacy-preserving model generation* or are primarily concerned with *privacy-preserving model serving*. As demonstrated in Section 2.1, the model inference computation tasks can be considered as a non-iterative and simplified form of model training procedures. Thus, from the standpoint of the computation, privacy-preserving inference problems could be considered as a subset of privacy-preserving training problems. The majority of PPML systems that emphasize privacy-preserving training via secure computation techniques also indicate theoretical support for privacy-preserving inference, such as in [87, 88, 89, 58, 64, 37]; these, therefore, may be regarded as full privacy-preserving pipeline approaches.

We emphasize that PPML solutions that rely on privacy-preserving data preparation techniques such as anonymization, sketching, or differential privacy are typically incompatible with privacy-preserving inference. The model inference aims at obtaining an accurate prediction for a single data point, those approximation or perturbation techniques are either inapplicable to the data required for prediction or lower the usefulness data for inference. Thus, the data-oriented PPML proposals as introduced in Section 3.1 are incompatible with the *privacy-preserving inference* goal.

Another direction of a full privacy-preserving pipeline could be simply integrating privacy-preserving model generation approaches and those privacy-preserving model serving approaches; for instance, we can integrate a privacy-preserving model query or publication-based method for model deployment with a secure computation based privacy-preserving inference). For instance, it is possible to produce

a deep neural networks model with the most privacy-preserving training approaches. Then the trained model can be transformed into an oblivious neural network to support privacy-preserving predictions.

4 Privacy Guarantee in PPML

Privacy, in general, is a broad term that encompasses the freedom of thought, control over one's body, seclusion in one's home, control over personal information, freedom from surveillance, protection of one's reputation, and protection from searches and interrogations [90]. Privacy, simply put, is a subjective estimate of the degree to which personal information can be revealed to untrustworthy domains/entities and how much personal information is publicly available. It's difficult to define what privacy is and how to measure it, because privacy is a subjective concept with varying opinions or points of view.

Typically, in the digital realm, a widely accepted minimum level of privacy protection is that of the personal *identity* [39, 45, 91]. Some common approaches to privacy include *differential privacy*, mechanisms [45, 91] and *k-anonymity* mechanism [39] and its follow-up work such as *l-diversity* [40], *t-disclosure* [41]. Specifically, differential privacy mechanisms try to conceal individuals from a dataset's output patterns, such as the output of specified functions queried from the dataset. The basic principle of differential privacy is that if the effect of an arbitrary single change in database entries is modest enough, the query result cannot be used to infer much about any specific individual, and thus provides privacy protection [45, 91]. The purpose of an anonymization procedure is to remove personally identifiable information directly from a dataset. Given a person-specific field-structured dataset, k-anonymity and its variations are dedicated to creating and concealing identifiers and quasi-identifiers in such a way that the individuals who are the subjects of the data cannot be re-identified while the data remain useful [39].

Numerous privacy-related terminologies and concepts are used in PPML proposals to determine their privacy-preserving capabilities, resulting in the absence of a standardized definition of privacy in PPML. We examine these commonly discussed privacy-related concepts in order to investigate PPML's privacy guarantees from two perspectives: *object-oriented* and *pipeline-oriented*. The former focuses on evaluating the privacy protection of specific objects in PPML, namely, the trained model weights, exchanged gradients, and training or inference data samples. The latter verifies the privacy assurance by evaluating the entire pipeline, as illustrated in Figure 2. Next, we expand on each perspective.

4.1 Object-Oriented Privacy Guarantee

The privacy claim of the majority of early PPML solutions is object-oriented, focusing on a single object such as a model or a data sample. A series of PPML solutions directly protect the dataset, such as by empirically deleting *identifiers* and *quasi-identifiers* from the dataset using anonymization mechanisms [39, 40, 41], to meet the privacy goal. To address the concerns raised by the above-mentioned privacy guarantee, a differential privacy (DP) mechanism [45, 91] has been developed and has been widely accepted across multiple domains as it provides a mathematically provable privacy guarantee. Additionally, encryption is a more rigorous approach to data protection, requiring learning from the dark because the data is encrypted. In the remainder of the paper, we generally refer to this type of privacy assurance as data-oriented privacy guarantee, or input privacy for short, as defined in Definition 1.

Definition 1 (Data Oriented Privacy Guarantee) *A PPML solution asserts the data-oriented privacy promise, which states that an adversary cannot learn private information directly from input training/inference data samples or associate private information with a specific person's identification.*

In short, data-oriented privacy-preserving approaches aim to prevent privacy leakage from the input dataset directly. However, privacy is not free; one unintended consequence of input privacy is the sacrifice of data utility. For instance, the anonymization mechanism needs to aggregate and remove proper feature values. Simultaneously, certain values of quasi-identifier features are erased altogether or in part to fulfill l-diversity and t-disclosure definitions. Additionally, a differential privacy technique requires the addition of a noise budget to the data sample. Both methods have

detrimental effects on the trained model's accuracy. While encrypted data may ensure the dataset's confidentiality, it brings extra processing burden to the subsequent machine learning training.

Another group of PPML solutions focuses on delivering privacy-preserving models, implying that the trained model in the PPML system is the privacy-preserving model. The privacy-preserving model's objective is to prevent privacy leaks in both the trained model and its use. Examples of privacy information may include include information related to membership, property, attribute, etc., of a subject in a given data sample. As stated in Definition 2, we refer to such a privacy assurance in the remainder of the paper as a *model-oriented privacy guarantee* or *model privacy* in short.

Definition 2 (Model Oriented Privacy Guarantee) *A PPML solution is said to provide a model-oriented privacy guarantee if and only if an adversary cannot derive any private information from a given model by querying it a number of times.*

Existing PPML approaches address model privacy guarantee using two approaches: (i) by incorporating differentially private training algorithms to perturb the trained model parameters; and (ii) regulating the model access times and model access patterns to limit the adversary's ability to get private information. For instance, Adabi et al. [36] propose a differentially private stochastic gradient descent (DP-SGD) algorithm by adding a differential privacy budget (noise) into the clipped gradients to achieve a differentially private model. The private aggregation of teacher ensembles (PATE) framework [78] creates an novel model deployment architecture in which a collection of ensemble models is trained as teacher models to offer model inference service for a student model.

4.2 Pipeline-Oriented Privacy Guarantee

Existing privacy measurement approaches such as differential privacy and k-anonymity are only applicable for certain entities such as data samples and trained models but cannot be directly adopted to assess the privacy guarantee of the entire PPML pipeline. There is a lack of formal or informal approaches for assessing the strength and scope of privacy protection provided by an ML pipeline. We argue that assessing the privacy guarantee relies on defining (i) the *boundary of data processing* and (ii) the *trust assumption on each processing domain* in the pipeline. For instance, suppose that a data owner employs a third-party computational facility (CF) to process its privacy-sensitive data; this creates a boundary in data processing workflow into two parts: data owner's local domain and CF's domain. If the data owner completely trusts CF, privacy concerns may not arise; otherwise, there is a need for a privacy guarantee with regards to data processing.

As illustrated in Figure 2, we establish the trust boundaries of the processing pipeline in PPML as *data producers*, *local CF*, *third-party CF*, and *model consumers*. From the perspective of a data owner (i.e., data producer), it may have varying levels of confidence in other domains. For instance, the data producer may fully trust its local CF, or have semi-trust on the third-party CF, or may have no trust at all in the model consumer. Based on such trust assumptions for each boundary, we present the taxonomy of privacy guarantees from the data owner's perspective as follows:

- *No Privacy Guarantee*: Here, the raw training data is shared with third-party CFs, regardless of their trustworthiness, and without using any privacy-preserving approaches. Each entity is able to acquire the original raw data to process or the ML model to consume according to its role in the ML pipeline.
- *Global Model Privacy Guarantee*: Global privacy guarantee focuses on model serving phase. A data producer generates a trained model using its own CFs or enlists the assistance of a *trustworthy* third-party entity with powerful CFs to assist in training the machine learning model using the raw data provided by the *data producer*. The global privacy guarantee is designed to prevent the leakage of sensitive information during the model deployment and inference phases. In essence, the machine learning model is a statistical abstraction or pattern generated from the raw data, and hence any privacy leakage that occurs is considered as a statistical leakage. Typical types of privacy leakage include the disclosure of membership, class representatives, and properties. For instance, a machine learning model for assisting in the diagnosis of HIV can be trained using current HIV patient healthcare records. A successful membership attack on the model enables an adversary to determine whether or not a specified target sample (i.e., patient record) was used in the training, hence disclosing whether or not a target person is an HIV patient.

- *Vanilla Local Privacy Guarantee*: The basic local privacy guarantee ensures that the privacy-sensitive raw data is not directly shared with other *honest* entities in the ML pipeline. The indirect-sharing approaches are as follows: (i) the raw data is pre-processed to remove privacy-sensitive information, or to obfuscate private information with noise before sending it out for model training; (ii) the raw data is pre-trained in a local model, with the generated model update being revealed to other entities.
- *Primary Local Privacy Guarantee*: The primary local privacy is built upon the vanilla local privacy guarantee. The primary distinction is in the trust assumption used to configure the remainder of the pipeline. Apart from the basic requirement of vanilla privacy guarantee, it also requires that the shared local model update should be protected from *curious* third-party CF entities; here *honest* CF entities may include the training server in the IaaS platform, and coordinating server in a distributed collaborative ML system.
- *Enhanced Local Privacy Guarantee*: The enhanced local privacy is built upon the primary local privacy guarantee by changing the assumptions to inlcude the third-party CF entities that are totally *untrusted*.
- *Full Privacy Guarantee*: The requirement of a full privacy guarantee includes both *local privacy* and *global privacy*. As the definitions as mentioned above, the *global privacy* guarantee focuses on the ML model serving phase, while the *local privacy* ensures the privacy guarantee in the model generation phase, the *full privacy* ensures privacy protection at each step in the ML pipeline, as illustrated in Figure 2.

In particular, the privacy leakage is more specific to the threat models considered that considers an adversary's behaviors and capabilities and assumes the worst-case scenario that a machine learning system can handle. Simultaneously, the threat model also reflects users' confidence in the entire data processing pipeline and the trustworthiness of each entity. As a result, as indicated previously, the privacy guarantees are highly correlated with the PPML system's specific threat model.

5 Technical Utility in PPML

We begin this section by classifying and discussing the PPML solutions in more detail by dissecting those solutions and examining how those approaches address the following questions:

- How the privacy-sensitive data is released or published?
- How the privacy-sensitive data is used for model training?
- Does the architecture of the ML system prevent the disclosure of private-sensitive information?

As such, we summarize the privacy-preserving approaches into four categories: *data publishing-based, data processing-based, architecture-based,* and *hybrid approaches* that integrate two or three of those approaches. Following that, we analyze the potential impact of these privacy-preserving techniques on normal ML solutions in terms of various utility costs, such as, *computation utility*, *communication utility*, *model utility*, *scalability utility*.

5.1 Type I: Data Publishing Approaches

In general, the data publishing based privacy-preserving approaches in the PPML system fall into the following categories: approaches that *totally eliminate* the identifiers and/or *partially conceal* quasi-identifiers in the raw data; approaches that *perturb* the statistical result of the raw data; approaches that *completely transform* the raw data through the use of confusion and diffusion techniques.

5.1.1 Elimination-based Approaches

The traditional anonymization mechanisms are classified as elimination-based approach to prevent privacy leakage, in which techniques such as k-anonymity[39], l-diversity[40] and t-closeness [41] are applied to the raw privacy-sensitive data in order to eliminate private information. Specifically, the k-anonymity mechanism aims to ensure that privacy sensitive information about an individual is indistinguishable from at least k-1 other individuals. To accomplish this, k-anonymity defines the

identifiers and *quasi-identifiers* for each data attribute, after which the identifiers are removed and the quasi-identifiers are partially obscured. The l-diversity mechanism is based on k-anonymity by additionally maintaining the diversity of sensitive field, namely, *equivalence class*. An equivalence class has l-diversity if it has at least l "well-represented" values for any privacy-sensitive attribute. Essentially, as an extension of the k-anonymity mechanism, the l-diversity mechanism reduces the granularity of the data representation while maintaining the variety of sensitive fields through the use of techniques such as generalization and suppression, where any record can be mapped to at least $k-1$ other records in the dataset. The t-closeness technique further refines the notion of l-diversity by imposing additional constraints on the value distribution on the *equivalence class*; here, an equivalence class has t-closeness if the distance between the distribution of a sensitive attribute in this class and the distribution of the attribute in the entire dataset is less than a threshold t.

Examples of emerging elimination-based PPML solutions include approaches proposed by Yang et al. and Ong et al. in [42, 92] that focus on secure or privacy-preserving federated gradient boosted trees model. Yang et al. in [42] employ a modified k-anonymity based data aggregation method to compute the gradient and hessian by projecting original data in each feature to avoid privacy leakage, instead of directly transmitting all exact data for each feature. Additionally, Ong et al. in [92] propose an adaptive histogram-based federated gradient boosted trees by a data surrogate representation approach that is compatible with either the k-anonymity method or differential privacy mechanism.

5.1.2 Perturbation-based Approaches

We discuss two commonly used perturbation-based approaches: differential privacy mechanisms and sketching techniques.

Differential Privacy: Typically, the perturbation-based privacy-preserving data publishing approaches primarily refer to (ϵ, δ)-*differential privacy* technique [45, 46, 47] and more recent (α, ϵ)- *Rènyi differential privacy* [93] that is based on Rényi divergence. According to [91, 46], differential privacy is formally defined as follows: a randomized mechanism $\mathcal{M}: \mathcal{D} \rightarrow \mathcal{R}$ with domain $\mathcal{D}$ and range $\mathcal{R}$ satisfies (ϵ, δ)-*differential privacy* if for any two adjacent input $d, d^{'} \in \mathcal{D}$ and for any subset of outputs $S \subseteq \mathcal{R}$, it holds that

$$\Pr[\mathcal{M}(d) \in S] \leq e^{\epsilon} \cdot \Pr[\mathcal{M}(d^{'}) \in S] + \delta. \tag{6}$$

The additive term δ allows for the possibility that plain ϵ-differential privacy is broken with probability δ (which is preferably less than $1/|d|$). Suppose that the Rényi divergence of order $\alpha > 1$ is defined as $D_{\alpha}(P||Q)$ over two probability distributions P and Q. Similarly, the Rènyi differential privacy requires the following condition is hold:

$$D_{\alpha}(\mathcal{M}(d)||\mathcal{M}(d^{'})) \leq \epsilon. \tag{7}$$

Usually, a paradigm of an approximating a deterministic function $f: \mathcal{D} \rightarrow \mathbb{R}$ with a differentially private mechanism is via *additive noise* calibrated to function's *sensitivity* S_f that is defined as the maximum of the absolute distance $|f(d) - f(d^{'})|$. The representative and common additive noise mechanisms for real-valued functions are Laplace mechanism ($\text{Lap}(\mu, b)$) and Gaussian mechanism ($\mathcal{N}(\mu, \sigma^2)$), as respectively defined as follows:

$$\mathcal{M}_{\text{Gauss}}(d; f, \epsilon, \delta) = f(d) + \mathcal{N}(\mu, \sigma^2) \tag{8}$$

$$\mathcal{M}_{\text{Lap}}(d; f, \epsilon) = f(d) + Lap(\mu, b) \tag{9}$$

The typical usage of differential privacy in the PPML solutions falls into two directions: (i) directly adopting the aforementioned additive noise mechanism on the raw dataset in the case of publishing data, as illustrated in [94, 95]; or (ii) transforming the original training method into a differentially private training method so that the trained/published model provides ϵ-differential privacy guarantee, as illustrated in [36, 52, 53].

Sketching: *Sketching* is an approximate and simple approach for data stream summarization, by empoying a probabilistic data structure that serves as an event frequency table, similar to counting Bloom filters. Recent theoretical breakthroughs, such as in [96, 97], have demonstrated that with some adjustments, differential privacy is achievable through sketching techniques. For instance, in [98], Balu and Furon focus on privacy-preserving collaborative filtering, a popular technique for the recommendation system, by utilizing sketching techniques to implicitly provide differential privacy

guarantees by leveraging advantage of the inherent randomness of the data structure. Recently, Li et al. in [43] propose a novel sketch-based framework for distributed learning, where they compress the transmitted messages via sketches to achieve communication efficiency and provable privacy benefits simultaneously.

In short, the traditional anonymization mechanisms and perturbation-based approaches are designed to tackle general data publishing problems; however, those techniques are still not out-of-date in the domain of PPML. The differential privacy mechanism, in particular, has been widely utilized in contemporary privacy-preserving deep learning and privacy-preserving federated learning systems, such as those proposed in [36, 51, 52, 53, 57, 35]. Additionally, differential privacy is demonstrated not just as a privacy-preserving approach, but also in the generation of synthetic data [45, 99, 100] and emerging generative adversarial networks (GAN) [101, 102].

5.1.3 Confusion-based Approaches

The confusion-based approach primarily refers to the *cryptography* technique that *confuses* the raw data in order to achieve a significantly greater privacy guarantee (i.e., confidential-level privacy) than typical anonymization mechanisms and perturbation-based approaches. Existing cryptographic approaches for data publishing for ML training fall into following two directions: (i) utilizing traditional symmetric encryption schemes such as AES in conjunction with the garbled-circuits and oblivious transfer to achieve general secure multi-party computation protocols [103, 104, 105] and (authenticated) encryption in conjunction with the pairwise masking techniques [38]; (ii) utilizing advanced modern cryptosystems such as homomorphic encryption schemes [29, 65, 66, 67, 68] and functional encryption schemes [30, 69, 70, 71, 72, 73, 74] that contain the necessary algorithms to compute over the ciphertext, such that one party with the issued key is able to acquire the computation results. The typical PPML system, such as those proposed in [88, 106], could be classified as the first direction of the crypto-based data publishing approach, whereas more recent studies, such as those presented in [57, 58, 35, 59, 60, 61, 62, 63, 64], emphasize on the direction (ii).

The confusion-based data publishing approaches (i.e., cryptographic-based systems), in particular, cannot work independently and are frequently coupled with subsequent secure process approaches, as the data receiver is meant to learn only the result of the data processing, rather than the raw data. Although confusion-based approaches are promising candidates for data publishing, their introduction and discussion should focus on how to share the one-time symmetric encryption keys in direction (i) or how to process the encrypted data in direction (ii). The following section will go into extensive detail.

5.2 Type II: Data Processing Approaches

The data processing approaches for training and inference are classified into two categories based on their respective data publication methodologies: *ordinary computation* and *secure computation*. As with *Type I* approaches discussed in Section 5.1, if the data is published using traditional anonymization mechanisms or perturbation-based approaches, in which personal identifiers in the data are eliminated and the statistical result is perturbed by adding differential privacy noise or constructing a probabilistic data structure, the consequent training computation is as normal as the training computation in vanilla machine learning systems. Thus, the privacy-preserving data processing refers primarily to the secure computation that performed throughout the training and inference phases.

Andrew Yao [107] initialized the secure computation problems and their accompanying solutions in 1982 using a garbled-circuits protocol for two-party computation challenges. The major purpose of secure computation is to enable two or more parties to evaluate an arbitrary function of both their inputs without revealing anything except the function's output to either side. According to the number of players enrolled, these secure computation approaches can be classed as basic secure two-party computation (2PC) and secure multi-party computation (MPC or SMC). From the threat model's (a.k.a., security model's or security guarantee's) perspective, such secure computation protocols provide two distinct levels of security in response to varied adversary settings: *semi-honest* (passive) security and *malicious* (active) security. We recommend the reader to [108, 109] for a detailed systematization of knowledge on secure multi-party computation solutions in general and their corresponding threat models.

We prefer to explore existing secure computation approaches in PPML in terms of the underlying technological principles in this section. Generally, these secure computation approaches fall under the following categories:

(i) additive blindness with perturbation, DC-net, or (verifiable) secret sharing;

(ii) garbled-circuits technique with the oblivious transfer;

(iii) modern advanced cryptography schemes;

(iv) mixed-protocols approaches;

(v) trusted execution environment technique with oblivious methods.

The underlying supported function could be generic or specific among various types of secure computation solutions. The remainder of the paper will elaborate on each category of the solution.

5.2.1 Additive Mask based Approaches

A subcategory of secure computing approaches is additive blindness (or masking) techniques based on perturbation, DC-net, or secret sharing, in which private data are masked with randomized values that can be canceled out in the final computation output.

Additive perturbation is a straightforward type of additive masking. For instance, in a generic additive perturbation based privacy-preserving summation [110] - a function-specific (i.e., the aggregation function) secure multi-party computation - the coordinator provides its input x_0 with adding a randomized perturbation r, and then each participant adds its input x_i on the $x_0 + r$ and passes to the next participant. Finally, the coordinator receives the $\sum x_i + r$ and eliminate its randomized perturbation r in order to obtain the aggregated result. Another sort of additive masking is multiplicative perturbation, in which values are perturbed using random projection or random rotation techniques.

Additionally, in pairwise additive masking-based secure computing approaches, various secret sharing techniques such as t-of-n secret sharing or multi-secret sharing are used. For instance, assume a group of individuals desires to collaboratively compute the sum of their individual inputs, with the assistance of a semi-trusted entity (called coordinator). For simplicity, we can assign the coordinator with a randomized nonce s as the secret. Then each participant is issued with the secret sharing s_i to add it into its input as a perturbation. Finally, the coordinator can sum the $\sum x_i$ by removing the recovered secret s. Recently proposed double-masking pairwise-based protocols [38, 111, 112] address participant failures by requiring pairs of participants to initially agree on pairwise masks via key exchange mechanisms. Following that, each participant adds a self-mask and the sum of the pairwise masks of the other participants to its input. In the recovery phase, the coordinator requests the alive participants with the sum of their (uncancelled) pairwise masks for the dropped users with added their "self-mask", and then subtracts those values from the previously masked input. As a result, it can correctly compute the sum of the inputs of the undropped participants.

Additionally, anonymous communication could be a viable solution for pairwise blinding-based approaches. For instance, dining cryptographer networks (DC-nets) [113] or mix-nets [114] are a sort of anonymous communication network in which only one person can send an anonymous message at a time. In general, anonymous communication can be viewed as a restricted case of secure aggregation. By utilizing DC-nets or mix-nets, the trusted coordinator can gather input from each party anonymously and then compute the function results, which provides a measure of privacy protection due to the coordinator's inability to determine the source of each function input.

In summary, the majority of additive blinding-based technique can be termed as lightweight approach in comparison to other forms of secure computing. As illustrated in proposals [115, 116, 116, 117, 118, 38], those secure computing approaches are widely adopted in the traditional data mining area [115, 116] and lack attention in the recent privacy-preserving machine learning proposals. Specifically, works such as demonstrated in [117, 118] focus on the k-means clustering machine learning algorithms. In [117] Bunn and Ostrovsky propose two types of additive blinding methods, namely, a *division protocol* and a *random value protocol* to perform two-party division and to sample uniformly at random from an unknown domain size. Doganay et al. [118] utilize additive secret sharing as a cryptographic primitive to implement a secure multiparty computation protocol for privacy-preserving clustering. Recent proposals, such as in [38, 111, 112], usually rely on a set of cryptographic primitives. They employ a t-of-n secret sharing scheme with additional DDH-based

Table 1: Illustrate of the garbled table for AND gate g_{AND}.

b_i	b_j	g_{AND}	encrypted output	permutation	garbled output
0	0	0	$\mathsf{Enc}_{w_i^0, w_j^0}(w_k^0)$	$\Rightarrow$	$\mathsf{Enc}_{w_i^0, w_j^0}(w_k^0)$
1	0	0	$\mathsf{Enc}_{w_i^1, w_j^0}(w_k^0)$	$\Rightarrow$	$\mathsf{Enc}_{w_i^1, w_j^0}(w_k^1)$
0	1	0	$\mathsf{Enc}_{w_i^0, w_j^1}(w_k^0)$	$\Rightarrow$	$\mathsf{Enc}_{w_i^0, w_j^1}(w_k^0)$
1	1	1	$\mathsf{Enc}_{w_i^1, w_j^1}(w_k^1)$	$\Rightarrow$	$\mathsf{Enc}_{w_i^1, w_j^1}(w_k^0)$

key agreement and authenticated encryption to construct a protocol for securely computing sums of vectors with low communication overhead, robustness to failures, and which requires only one server with limited trust. To further improve computation and communication efficiency, Turbo-Aggregate [112] employs additive secret sharing and a multi-group circular strategy for secure aggregation tasks. Simultaneously, FastSecAgg [111] proposes a novel multi-secret sharing scheme based on a finite-field version of the Fast Fourier Transform technique.

5.2.2 Garbled Circuits based Approaches

The garbled circuits and oblivious transfer techniques serve as the foundation for constructing another type of secure computing solutions. To exemplify this, we utilize the two-party secure computation (2PC) protocol here. The fundamental idea of 2PC is that one party (referred to as the garbled-circuit generator) creates a circuit computing function that includes a large number of garbled gates encrypted using typical symmetric encryption algorithms such as AES. Then the other party (a.k.a, the garbled-circuit *evaluator*) computes the output of the circuit obliviously, without learning any intermediate information. Specifically, the function f is transferred to a Boolean circuit comprised of vast amounts of garbled gates in various sorts (e.g., AND-gate, OR-gate, and XOR-gate). Suppose that an AND-gate g^{AND} is associated with two input wires i and j, and one output wire k. The generator first generates two cryptography keys for each input wire, denoted as $w_i^0, w_i^1, w_j^0, w_j^1$, where the superscript represents the the encoded input bits (e.g., w_i^0 encodes 0-bit input of wire i, while w_i^1 encodes 1-bit input of wire i). For inputs data $b_i, b_j \in \{0, 1\}$, the *generator* computes the ciphertext as $\mathcal{E}^{\text{symmetric}}.\mathsf{Enc}(w_k^{g^{\text{AND}}(b_i, b_j)})$ with keys $w_i^{b_i}, w_j^{b_j}$. Table 1 presents the gate table in detail. Then, the *evaluator* is able to acquire its input wire associated keys w_j^0, w_j^1 with its input $b_j \in \{0, 1\}$ without revealing that input to the *generator* using the 1-of-2 oblivious transfer (OT) technique. With the input associated key w^{b_j} and the received permuted garbled table, the *evaluator* is able to decrypt the corresponding ciphertext to acquire the output $w_k^{g^{\text{AND}}(b_i, b_j)}$ without learning the input of the *generator*. Finally, those different types of garbled gates can compose any functions used in the secure computation protocols.

Even though garbled-circuits based 2PC and MPC problems is not an emerging topic and have been studied for over 40 years, the security community continues to work on them and attempts to improve their efficiency and practicality [103, 119, 120, 104, 121]. As a result of these efforts, the garbled-circuits based 2PC or MPC has been recently adopted to address the challenge of secure computation issues in popular machine learning algorithms, and more specifically, complex deep learning models [59, 87, 81, 88, 122]. For example, Chameleon [106] combines the best aspects of generic secure function evaluation protocols, where it employs additive secret sharing values to achieve linear operations and garbled-circuit protocols to implement nonlinear operations. Similar to the Chameleon framework, ABY^3 [123] proposes and implements a general framework for diverse machine learning algorithms in a three-server paradigm, based on the mixed 2PC presented by Demmler et al. [124], wherein data owners secretly share their data among three servers who train and evaluate models on the joint data using three-party computation. Notably, several of those solutions, such as Chameleon and ABY^3, also make use of the homomorphic encryption technique, which will be discussed in later in this section.

DeepSecure [88], which is still based on Yao's garbled circuits, is a secure deep learning framework supporting various types of nerual networks that is built on automated design, efficient logic synthesis, and optimization methodologies. Additionaly, Riazi et al. [125] propose another end-to-end framework based on Yao's protocol that supports a paradigm shift in the conceptual and practical realization of privacy-preserving inference on deep neural networks. DeepSecure's protocol is optimized for

discretized neural networks with integer weights, whereas XONN is tuned for binary neural networks with boolean weights. These quantized networks boost performance by eschewing expensive fixed-point multiplication in favor of integer or binary multiplication. Recently, Agrawal et al. [126] proposed QUOTIENT, a novel method for discretized DNN training along with a customized 2PC protocol. EzPC [122] is another sort of 2PC framework that generates efficient 2PC protocols from high-level, easy-to-write programs. To achieve performance improvement, the proposed compiler of EzPC generates protocols0 combining both arithmetic and boolean circuits techniques.

Without relying on non-colluded two or three servers, those garbled-circuits-based solutions can provide provably security guarantee and demonstrate their promises in the training phase rather than merely the inference phase for deep neural networks, as exemplified in [88, 126]. Those systems, however, suffer from transmission overhead. As illustrated in Table 1, to perform a simple computation on two input bits such as $b_i \wedge b_j$, it is mandatory to transmit a set of ciphertexts of fixed size and an additional oblivious transmission overhead for key delivery, where the size of each ciphertext and key depends on the secure parameter of the chosen symmetric encryption scheme. Then, when complex computation functions such as those used in machine learning are considered, the size of transferred data explodes substantially.

5.2.3 Advanced Cryptographic Approaches

Another important direction of building secure multi-party computation is the modern cryptographic approaches, which mostly refer to advanced cryptosystems such as homomorphic encryption and functional encryption that enable computation over the ciphertext. Modern advanced cryptosystem-based secure computation can achieve a high level of privacy guarantee, as does the garbled-circuits-based approach, which makes use of the cryptosystem to provide confidentiality-level privacy. Unlike garbled-circuits-based secure protocols, which are constrained by enormous amounts of transmitted data, modern advanced cryptosystem-based approaches require only encrypted data to be transferred, instead of the data-encoded garbled-circuits and corresponding keys via oblivious transfer technique. Here we briefly introduce the *homomorphic encryption* schemes [29, 65, 66, 67, 68] and *functional encryption* schemes [30, 69, 70, 71, 72, 73, 74] that are primarily employed in existing PPML proposals [58, 127, 128, 129, 62, 63, 64, 130, 63].

Homomorphic Encryption (HE) is a public-key cryptosystem with the capacity of computing over ciphertexts without access to the private secret key. The result of the computation over the ciphertexts remains in the form of ciphertext. Simultaneously, the decrypted result corresponds to the outcome of operations performed on the original plaintext. According to the capabilities of performing various kinds of operations, typical HE types include *partially* homomorphic, *somewhat* homomorphic, *leveled fully* homomorphic, and *fully* homomorphic encryption. Unlike traditional public-key schemes, which consis of three primary algorithms: key generation (*KGen*), encryption (*Enc*), and decryption (*Dec*), an HE scheme has an additional *evaluation* (*Eval*) algorithm for performing operations over ciphertext according to specified functions. Formally, a HE scheme $\mathcal{E}_{\text{HE}}$ includes the preceding four algorithms such that

$$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathcal{E}_{\text{HE}}.\mathsf{KGen}(1^{\lambda}) \tag{10}$$

$$C_{\text{HE}} \leftarrow \{\mathcal{E}_{\text{HE}}.\mathsf{Enc}_{\mathsf{pk}}(m_i)\}_{i\in\{1,\ldots,n\}} \tag{11}$$

$$C^{f}_{\text{HE}} \leftarrow \mathcal{E}_{\text{HE}}.\mathsf{Eval}_{\mathsf{pk}}(f, C_{\text{HE}}) \tag{12}$$

$$f(m_1, ..., m_n) \leftarrow \mathcal{E}_{\text{HE}}.\mathsf{Dec}_{\mathsf{sk}}(C^{f}_{\text{HE}}) \tag{13}$$

where $\{m_1, ..., m_n\}$ are the message to be protected, pk and sk are the key pairs generated by the key generation algorithm.

Typically, two representative approaches are available for achieving generic secure computation via HE techniques: *preprocessing-model* approach and *pure fully homomorphic encryption* (FHE) approach. The former approach presupposes a trustworthy dealer, who does not need to know the function to be computed or the inputs and can be implemented through secure protocol with public-key infrastructure, but simply offers raw materials for the computation. Additionally, these operations can be performed as part of a preprocessing step using somewhat homomorphic encryption (SHE) techniques. Following that, the online protocol evaluates a function securely by utilizing only low-cost information-theoretic primitives. The *pure FHE* approach, derived from the approach of FHE by Gentry [29], is more straightforward than the *preprocessing model* approach. In a pure FHE

approach, all parties first encrypt their input using the FHE scheme and then use the homomorphic properties of the ciphertexts to evaluate the desired function on them. Following that, these parties can conduct a distributed decryption operation on the final ciphertexts to obtain the results.

Here, instead of elaborating on all HE achievements, we briefly introduce commonly employed HE implementations shown in existing PPML systems. We direct the reader to the article presented by Acar et al. [68] for the theory and implementation survey on HE schemes, as well as to an open consortium of HE standardization [131] to examine the availability of open-source libraries, for additional information. The Paillier cryptosystem [132], an additive homomorphic (partially homomorphic) encryption scheme, is one of the most extensively used HE implementations. Given the message m_i and m_j, the Paillier system $\mathcal{E}_{\text{HE}}^{\text{Paillier}}$ supports the additive homomorphic operation such that

$$\mathcal{E}_{\text{HE}}.\mathsf{Enc}(m_i) \circ \mathcal{E}_{\text{HE}}.\mathsf{Enc}(m_j) = \mathcal{E}_{\text{HE}}.\mathsf{Enc}(m_i + m_j) \tag{14}$$

The HElib [133] implemented various well-known fully homomorphic encryption schemes, including [66, 77, 134, 135], while also incorporating optimization techniques like as bootstrapping, smart-vercauteren, and approximate number. SEAL [136] is another HE library that enables the computation of additions and multiplications on encrypted integers or real numbers. However, other operations, such as encrypted comparison, sorting, and regular expressions, are rarely possible to be evaluated on encrypted data using this library. PALISADE [3] is a more contemporary and general lattice cryptography library that presently supports efficient implementations of the following lattice cryptography capabilities including FHE schemes such as BGV [66], CKKS [77], as well as multi-party extensions of FHE [137].

Several early proposals for privacy-preserving machine learning incorporate HE into regular machine learning models to prevent privacy leaking. For instance, Hall et al. use homomorphic encryption to create a secure protocol for regression analysis in [127]. Simultaneously, Nikolaenko et al. [128] concentrate on privacy-preserving ridge regression on millions of records through the use of homomorphic encryption and garbled circuits. Additionally, Cock et al. [129] present a computationally secure two-party protocol that is based on additive homomorphic encryption and eliminates the need for a trusted initializer. Chialva and Dooms [138] recently attempted to analyze the feasibility of homomorphic encryption being completely implemented in machine learning applications by addressing the comparison and selection/jump operations challenges.

HE is being used in conjunction with the success of emerging deep neural networks to achieve privacy-preserving deep learning. For example, Gilad-Bachrach et al. [62] propose *CryptoNets*, which aims to deploy neural networks over encrypted data by employing a leveled homomorphic encryption scheme to the training data. *CryptoNets* enable the addition and multiplication of encrypted data but require prior knowledge of the arithmetic circuit's complexity. In contrast to the probable ineffectiveness of deeper neural networks in *CryptoNets*, Chabanne et al. [63] incorporate the batch normalization principle for the classification task using *CryptoNets*' fundamental ideas. Mishra et al. [139] recently proposed the Delphi framework for a cryptographic inference service for neural networks. They did so by constructing a hybrid cryptographic protocol that reduces communication and computation costs in comparison to previous work, as well as by developing a planner that generates neural network architecture configurations automatically. As with Delphi, Lehmkuhl et al. [140] propose the Muse framework to handle the challenge of fully malicious clients in secure inference scenarios rather than semi-honest clients. Zheng et al. [141] present *Helen*, a framework for maliciously secure *coopetitive* learning of a linear model without disclosing their data during the process of a distributed convex optimization technique called alternating direction method of multipliers (ADMM), in which a generic maliciously secure multi-party computation is based on the SPDZ protocol [142] derived from SHE schemes. Following that Alexandru et al. [143] concentrate on encrypted distributed Lasso for sparse data predictive control using ADMM, which ensures the computational privacy of all data, including intermediate outcomes. Additionally, Zheng et al. [144] present Cerebro, an end-to-end learning platform that addresses the trade-off between privacy and transparency, as well as the trade-off between generality and performance. Chen et al. [145] present a multi-key homomorphic encryption scheme with packed ciphertexts and demonstrate how it can be used to securely evaluate a pre-trained convolutional neural network (CNN) model, in which a cloud server provides online prediction services to a data owner using a classifier provided by a model provider, while maintaining the privacy of both the data and the model. Furthermore, Nandakumar et al. [64] enable the secure training over deep neural networks using the open-source FHE toolbox

[3] https://palisade-crypto.org/

HElib via a stochastic gradient descent training method. Several more recent publications, such as [146, 130, 147], focus on the same problem but employ a variety of optimization techniques to improve model efficiency and accuracy.

Functional Encryption (FE) is another type of public-key cryptosystem that enables computation over the ciphertext. Generally, a FE scheme $\mathcal{E}_{\text{FE}}$ consists of four algorithms: *setup*, *key generation*, *encryption* and *decryption* algorithms such that

$$(\mathsf{pk}, \mathsf{msk}) \leftarrow \mathsf{Setup}, \tag{15}$$

$$(\mathsf{sk}_f) \leftarrow \mathsf{KGen}(f, \mathsf{msk}), \tag{16}$$

$$C_{\text{FE}} \leftarrow \{\mathcal{E}_{\text{FE}}.\mathsf{Enc}_{\mathsf{pk}}(m_i)\}_{i\in\{1,\ldots,n\}}, \tag{17}$$

$$f(m_1, ..., m_n) \leftarrow \mathcal{E}_{\text{FE}}.\mathsf{Dec}_{\mathsf{sk}_f}(C_{\text{FE}}), \tag{18}$$

where $\{m_1, ..., m_n\}$ are the messages to be protected; the Setup algorithm creates a public key pk and a master secret key msk, and KGen algorithm uses msk to generate a new functional private key sk_f associate with the functionality f. Typically, the algorithms of *Setup* and *KGen* usually are run by a trusted third-party authority.

Regarding implementations of FE, except for CiFEr[4], PyFE[5], and FE-related PPML open-source projects: NN-EMD[6] and Reading-in-the-Dark[7], there is a dearth of well-known implementation libraries comparing to HE-related libraries such as HElib and SEAL. Existing construction of functional encryption schemes for general functionality, such as those recently proposed in [69, 148, 149, 150, 151, 152], place a premium on theoretical feasibility or presence of functionality. Only a few recent proposals, for example in [70, 75, 153, 154], emphasize the simplicity and applicability of FE, although the functionality is limited to the inner-products.

As mentioned previously, the primary similarity between the *FE* and *HE* is that they both permit computation over the ciphertext. The primary distinction between functional and homomorphic encryption, at a high level, is who can obtain the disclosed computation result. Given an arbitrary function $f(\cdot)$, homomorphic encryption allows computing *an encrypted result of* $f(x)$ from an encrypted x. In contrast, functional encryption allows computing *a plaintext result of* $f(x)$ from an encrypted x [155]. Intuitively, the function computation party in the HE scheme (i.e., the evaluation party) can only contribute its computation power to obtain the encrypted function result but cannot learn the function result unless it has the secret key. In contrast, the function computation party in the FE scheme (i.e., usually, the decryption party) can obtain the function result with the issued functional private key. Besides, except for most recently proposed decentralized FE schemes [156, 74, 157], the classic FE schemes are relied on a trusted third-party authority to provide key services, such as issuing a functional private key associated with specific functionalities. We recommend the reader to [158] for a thorough analysis of the unique properties of new and promising functional encryption approaches for a variety of secure computation workloads.

Unlike the widely used HE-based approaches for the PPML secure computation tasks, recently proposed FE-based PPML solutions, such as those in [35, 58, 159], are beginning to demonstrate their efficiency and applicability. Ryffel et al. [159] present a viable method for performing partially encrypted and privacy-preserving predictions using adversarial training and functional encryption. Using the FE to construct the secure computing mechanism, Xu et al. [58] initialize a CryptoNN framework that facilitates training a neural network model over encrypted data. Additionally, Xu et al. focus on privacy-preserving federated learning (PPFL) in [35], where they leverage the FE to design a secure aggregation technique that protects each participant's input in the PPFL.

Impact of Data Encoding. Unlike the computation in anonymized or deferentially private PPML training, which is performed in the same way as non-PPML computation over floating-point numbers, the secure computation in crypto-based PPML training should be performed in the integer format, as those cryptosystems are constructed in the integer group. As a result, there is a process for converting the data to integer format prior to doing the secure computation and then recovering the result in floating-point numbers. As a result of this procedure, a problem arises during cryptographically secure computation: how to determine the encoding degree and the resulting influence of encoding

[4] https://github.com/fentec-project/CiFEr

[5] https://github.com/OpenMined/PyFE

[6] https://github.com/iRxyzzz/nn-emd

[7] https://github.com/edufoursans/reading-in-the-dark

precision. As demonstrated in part in [58, 35, 160], the encoding issue is a trade-off problem, in which increased encoding precision implies increased model accuracy. In contrast, a higher encoding precision typically results in much more secure computation time (i.e., more training time, especially in the large scale of data training.)

5.2.4 Mixed-Protocol Approach

The mixed-protocols solution that combines the aforementioned techniques is another direction to achieve efficient and practical secure multi-party computation. The underlying principle of these mixed-protocol approaches is to evaluate computing operations in terms of their most efficient representations. The additions and multiplications with an efficient representation as an arithmetic circuit can use a homomorphic encryption approach. By contrast, the comparisons with an efficient representation as a boolean circuit will use Yao's garbled circuits technique.

Representative mixed-protocol solutions include TASTY [161], ABY [124], ABY^3 [123], Chameleon [106], CrypTen [162], Falcon [163], etc. Notably, while a portion of the solutions listed above were presented in Section 5.2.2 as the garbled-circuits approach, we revisit those frameworks in a mixed fashion here. Henecka et al. [161] present the *TASTY* compiler, which is capable of generating protocols using HE, efficient garbled circuits, and their combinations. Demmler et al. [124] propose the *ABY* framework, a mixed-protocol framework for efficiently combining secure computation approaches that are based on arithmetic sharing, boolean sharing, and garbled circuits. *ABY* pre-computes all cryptographic operations and then efficiently converts between secure computing approaches using pre-computed oblivious transfer extensions. Following that, Mohassel and Rindal then enhance the ABY framework and offer ABY^3 for privacy-preserving machine learning in a three-party environment [123]. Riazi et al. [106] propose the Chameleon framework to improve performance in terms of computation and communication between parties by overcoming two limitations: using a semi-honest third-party to preprocess arithmetic triples rather than the oblivious transfer used in ABY and handling signed fixed-point numbers.

Recently, in order to promote secure MPC adoption in the machine learning domain, Knott et al. [162] proposed the *CrypTen* framework, which exposes popular MPC primitives via abstractions that are commonly appeared in various machine learning frameworks, such as tensor computations, automatic differentiation, and modular neural networks. Li et al. [164] present *PrivPy*, an efficient framework for collaborative data mining with privacy protection, with the goal of providing an elegant end-to-end solution for data mining programming. *PrivPy*, in particular, provides more practical Python front-end interfaces, covering a broad range of functions frequently used in machine learning. Simultaneously, the core compute engine is built on secret sharing, provides efficient arithmetics, and supports SPDZ [142], and ABY^3 [123]. Notably, *PrivPy* does not provide a theoretical breakthrough in cryptographic protocols; instead, it creates a practical solution that enables elegant machine learning programming on mixed MPC frameworks while making the appropriate trade-offs between efficiency and security. As a follow-up to *PrivPy*, Fan et al. [165] focus on the privacy-preserving principal component analysis (PCA) via demonstrating an end-to-end optimization of a data mining algorithm to run on the mixed-protocol MPC framework. To further improve the efficiency, CRYPTGPU [166] is proposed to accelerate the mixed-protocol MPC computation via GPU. Specifically, CRYPTGPU introduces a new interface to losslessly embed cryptographic operations over secret-shared values into floating-point operations that highly-optimized CUDA kernels can process for linear algebra.

5.2.5 Trusted Execution Environment Approach

The trusted execution environment (TEE) is a technique for creating an isolated environment that operates on a separate kernel and incorporates security features such as code authentication, runtime state integrity, and confidentiality for its code, data, and runtime states kept in permanent memory. As a result, it is capable of providing a trusted environment in which users can run their programs in an untrusted server. Notably, TEE intends to reduce the trusted scope of computation resources from the owner (e.g., cloud computing provider) to the maker of computation facilities (e.g., CPU manufacturer). The TEE technique, in particular, is a method for achieving secure (or trust) computation for recently proposed work such as those in [167, 168].

The TEE approach is reliant on the presence of a secure hardware enclave. Examples of hardware enclaves include Intel SGX [169] and AME Memory Encryption [170]. Apart from trusted computing, another critical feature of TEE is remote attestation, which enables a remote client to properly verify

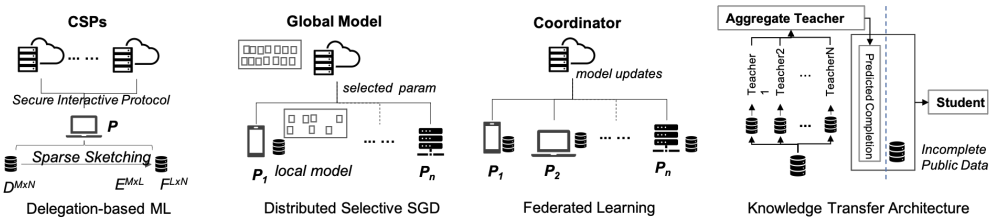

Figure 3: Representative architectures that have been incorporated into existing PPML solutions.

that certain software has been loaded into an enclave securely. Prior to that, a secure channel between the enclave and the client is bootstrapped, in which the enclave receives the client's public key and provides the signed attestation report.

A disadvantage of the TEE-based approach is the possibility of side-channel attacks that take advantage of information gleaned through the hardware implementation rather than flaws in the proposed algorithm itself. Example of exploited information in the side-channel includes timing information, power consumption, electromagnetic leaks or even sound that can provide an additional source of information. As a result, the TEE-based approach is frequently used in conjunction with oblivious techniques. Typically, the accessible memory addresses must be concealed by making the enclave execution oblivious to the secret data, which requires either employing an oblivious data structure [171] within the enclave or operating the enclave atop an ORAM [172]. For example, Law et al. [168] present a secure collaborative XGBoost system that enables multi-party training and inference of XGBoost models. Training takes place in the cloud, and individual clients' data is not revealed to the cloud environment or to other clients. Additionally, they modify XGBoost's algorithms to be data-agnostic to avoid potential side-channel risk. Similarly, Chamani and Papadopoulos [167] present two modified versions of *SecureBoost* [61] that address the partial privacy leakage induced by sample partial ordering when the active party or all parties have access to the TEE.

Except for the TEE-based secure boost model examples discussed above, Cheng et al [173] recently propose *Truda* that targets on a cross-silo FL system. *Truda* utilizes a decentralized and trustworthy aggregation architecture to alleviate information concentration around a single aggregator, in which all shared model updates in FL are disassembled at the parameter level and re-stitched to random partitions designated for multiple TEE-protected aggregators. As a result, each aggregator only has a fragmentary and shuffled view of model updates and is oblivious to the model architecture. Additionally, Zhang et al. [174] and Mo et al. [175] address the issue of scaling TEE-based distributed machine learning systems to large models or training datasets while remaining constrained by the capacity of the enclave page cache (EPC) or memory. Specifically, Zhang et al. [174] propose a TEE-based method to divide ML training into training and aggregating parts, making it possible to spin up a distributed cluster to accommodate voluminous multi-sourced data. Mo et al. [175] propose a PPFL solution, where both parties and aggregator own the TEE hardware for local training and global aggregation, respectively, by adopting greedy layer-wise training and aggregation to overcome the constraints posed by the limited TEE memory. As a result, it can provide comparable accuracy of complete model training but with the price of a tolerable delay.

5.3 Type III: Architectural Approaches

Recently, a portion of promising privacy-preserving machine learning solutions has been achieved through intentionally or unintentionally designed architecture. For example, PATE's architecture is primarily focused on knowledge transfer or knowledge distillation but also includes a guarantee of privacy for the disclosed model [78]. In short, there is no universally applicable design principle for privacy-preserving architecture-based approaches. Figure 3 illustrates representative architectures used in existing PPML solutions to demonstrate how privacy can be ensured by designing a privacy-aware architecture with appropriate trust assumptions and threat model settings. This section discusses representative architecture-related solutions for implementing PPML, rather than exhaustively listing all possible architecture-related privacy-preserving approaches.

5.3.1 Delegation-based ML Architecture

Delegation-based architecture is a classic architecture that gives the computation-limited parties the capability to create and use the ML models. Additionally, by incorporating additional secure techniques or making appropriate trust assumptions, the delegation-based architecture can provide privacy-preserving functionality for the machine learning system. Mirhoseini et al. [89], for example, propose CryptoML, a practical framework that enables provably secure and efficient delegation for contemporary matrix-based machine learning systems, in which a delegating client with memory and computational resource constraints can assign storage and computation to the cloud via an interactive delegation protocol based on provably secure Shamir's secret sharing. Similarly, Li et al. [176] propose a framework for privacy-preserving outsourced classification in a cloud computing environment, in which an evaluator can securely train a classification model using multiple encrypted data sources with distinct public keys. Rather than outsourcing computation to a single server, Mohassel and Zhang [87] present SecureML, an efficient two-server model in which data owners distribute their private data to two non-colluding servers that use 2PC to train various models on the joint data using stochastic gradient descent.

5.3.2 Distributed Selective SGD Architecture

Shokri and Shmatikov [177] propose a distributed selective SGD framework that enables multiple parties to jointly learn an accurate neural network model without sharing their input datasets. The system consists of several participants, each of whom has their own private training dataset, and one parameter server that is responsible for maintaining the most recent values of parameters available to participants. More precisely, the approach presupposes two or more people training concurrently and independently. For each round of local training, participants obtain the most recent values of the most-updated parameters and integrate them into local gradients using the selected partial parameters. Following the local training, each participant has complete control over which gradients, how many gradients, and how often they share with the parameter server.

5.3.3 Federated Learning (FL) Architecture

The FL [4, 5] is also a distributed machine learning framework with a similar architecture to the distributed selective SGD approach [177], in which each participant maintains a private local dataset of its mobile facilities and a coordinator (a.k.a., an aggregator or a central server, as used in a few papers) trains a shared global model using the local model updates generated by those participants. Due to the fact that training data does not leave the domain of each participant, FL can provide a primary level of privacy guarantee, as training data may be sensitive to sharing.

The FL design, in particular, can be viewed as a generic paradigm for the distributed selective SGD architecture outlined above. The FL framework requires that the participant downloads all parameters in the global model, trains it on a local dataset, and then uploads the full local model (update) to the coordinator server if the participant has not dropped out during the current training epoch. In comparison, the distributed selective SGD approach allows the participant more discretion over which partial parameters are included in the trained local or global models. The distributed selective SGD technique is, in essence, a sketching version of the FL.

Notably, FL has developed into a promising machine learning topic, attracting several studies on efficiency, scalability, security, and privacy. However, the growing systematization of knowledge article and survey have succinctly summarized FL, and as a result, we will not elaborate on FL here. For specifics, we direct the reader to [25, 178, 26, 28] .

5.3.4 Knowledge Transfer Architecture

The knowledge transfer-related architecture's major objective is to focus on knowledge distillation, model compression, and transfer learning. However, a portion of emerging knowledge transfer systems may include a guarantee of privacy.

The architecture of private aggregation of teacher ensembles (PATE) [78] and its variant [79, 80] are representative knowledge transfer-based PPML solutions. In general, the knowledge of an ensemble of teacher models (i.e., the models that were initially trained) is transferred to a student model (i.e., the model that will be used), with intuitive privacy provided by training teachers on disjoint data and strong privacy guaranteed by noisy aggregation of teachers' responses.

The architecture of private aggregation of teacher ensembles (PATE) [78] and its variant [79, 80] is a representative knowledge transfer based PPML solution. In general, the knowledge of an ensemble of "teacher" models (i.e., initially trained models) is transferred to a "student" model (i.e., model that will be used), with intuitive privacy provided by training teachers on disjoint data and strong privacy guaranteed by noisy aggregation of teachers' answers. In particular, PATE improves upon a specific, structured application of knowledge aggregation and transfer techniques. PATE specifically strengthens the guarantee of privacy by limiting student training to a limited number of teacher votes and revealing just the topmost vote after carefully adding random noise. Additionally, PATE restricts students' access to their teachers, allowing their exposure to teachers' knowledge to be quantified and bounded meaningfully utilizing knowledge transfer techniques such as generative adversarial networks (GANs).

In contrast to PATE's intuitive privacy, other widely used knowledge transfer techniques include *model transformation* and *model compression*. MiniONN [81] and variants [82], for example, turn an existing model into an oblivious neural network capable of making privacy-preserving predictions. Existing deep learning models in the natural language processing area with a large number of model parameters can be reduced to create lightweight deep learning models using knowledge distillation techniques [83, 84]. Apart from reducing the size of the deep learning model, as demonstrated and analyzed in [85, 86], it can also bring extra privacy-preserving functionalities. Recently, it has been demonstrated that model compression and neural network pruning can be employed to achieve privacy-preserving functionality [179, 180]. Huang et al. [179], for example, establish a link between neural network pruning and differential privacy. Additionally, Wang et al. [180] introduce the *DataLens* framework, where they describe a scalable privacy-preserving training approach based on gradient compression and aggregation. In comparison to PATE, which only allows ensemble teachers to vote on one-dimensional predictions, *DataLens* combines top-k dimension compression with a related noise injection method to enable voting on high-dimensional gradient vectors while maintaining privacy.

5.4 Type IV: Hybrid Approaches

Due to increased privacy protection requirements and recently demonstrated privacy threats such as membership inference attacks [11, 12, 13, 14, 15], model inversion attacks [16, 18, 181, 17], and deep gradient leaking [182, 22, 183], implementing a single type of privacy-preserving technique outlined above is insufficient in some cases. To achieve a higher level of privacy guarantee, an increasing number of well-built PPML systems incorporate more than one of the methodologies outlined above within an architecture that is appropriately tailored for the application scenario and threat model.

For example, the existing FL framework provides just a rudimentary guarantee of privacy, as each participant can save training data locally. The global trained model, however, cannot withstand these membership inference and gradient inference attacks. To solve this issue, as described in [36, 52, 79], one type of hybrid method attempts to blend architecture-based approaches and differential privacy mechanisms. Similarly, Liu et al. [184] propose exchanging sketched model updates rather than typical local model updates, as most FL systems do, because they recognize that sketching algorithms have the unique advantage of providing both privacy and performance benefits while retaining accuracy. Additionally, to avoid the curious coordinator investigating the participants' input in FL and to improve the global model performance, another type of hybrid approach, as proposed in [57, 35, 185], integrates crypto-based secure aggregation approaches and differential privacy mechanisms into the FL framework to provide a stronger privacy guarantee, where secure aggregation approaches of [57, 35, 185] are respectively based on partially additive homomorphic encryption, functional encryption, multi-key homomorphic encryption. Apart from the hybrid study of differential privacy and cryptography in privacy-preserving federated learning systems, a new comprehensive study [186] presents more examples of the conjunction of differential privacy mechanisms with cryptography schemes in generic domains. We refer the reader to [186] for a more extensive introduction divided into two categories: *differential privacy for cryptography* and *cryptography for differential privacy*.

Another type of hybrid method focuses on privacy guarantee during model training rather than final trained model via attempting to integrate the privacy-preserving architecture-based techniques and secure computation techniques. For example, recent works, as those in [60, 61], employ secure multi-party computation and FL approaches to distribute training of machine learning models across many

Table 2: Summary of primary technical path and utility cost of adopted privacy-preserving techniques in PPML solutions.

Techniques	Primary Technical Design	Utility Cost
k-anonymity (l-diferisty, t-closeness)	anonymize private information	model utility
differential privacy	perturb private information	model utility
sketching	sample from private information	model utility
compression	compress private information	model utility
homomorphic/functional encryption	diffuse/confuse private information	computation utility
pairwise additive mask (with SS, PKI) †	mask private information	communication utility
boolean garbled circuits (GC)	generic 2PC/MPC †	communication utility
boolean/arithmetic GC, SS, fully HE †	mixed 2PC/MPC †	communication utility
trusted execution environment	provide confidential computing	scalability utility
knowledge transfer architecture	prevent leakage from model	scenario utility
federated learning architecture	prevent data sharing	privacy strength utility

† Abbreviation: PKI - public key infrastructure; SS - secret sharing; 2PC - secure two-party computation; MPC - secure multi-party computation.

vertically partitioned datasets. Likewise, these PPML systems [89, 176, 87] employ delegation-based architectures and secure computing methodologies.

In short, classic anonymization mechanisms and perturbation techniques (e.g., differential privacy) can protect the final trained model from the majority of model inference attacks. Intuitively, the privacy-preserving approaches have obliterated or altered the private information contained in the training data samples; consequently, the final model cannot learn any privacy from the data. Furthermore, in a distributed delegation-based machine learning scenario or in a FL paradigm, those systems cannot completely avoid the honest-but-curious central server investigating the users' input during the training phase. Secure computation techniques such as garbled circuits or crypto-based 2PC and MPC can be used to protect the input of each participant. They cannot, however, prevent the final trained model from leaking private information.

Recently, the FL paradigm has been combined with the trusted execution environment (TEE) approach to create privacy-preserving FL [187, 188, 175, 173]. For example, Hashemi et al. [187] propose constructing secure enclaves within the coordinator server of the FL paradigm by utilizing a TEE. Each client can then encrypt and transmit their gradients to verifiable enclaves. Because the gradients are decrypted within the enclave, gradient-related privacy violations are avoided. Additionally, Zhang et al. [188] present ShuffleFL, a method for defending against side-channel attacks that combines random group structure and intra-group gradient segment aggregation. Mo et al. [175] address the issue of current TEEs having a limited memory space when employed in the FL by utilizing the greedy layer-wise training method to train each model's layer within the trusted area. Cheng et al. [173] propose *Truda*, a cross-silo FL system that relies on a decentralized and trustworthy aggregation architecture to alleviate information concentration around a single aggregator. All shared model updates in FL are disassembled at the parameter level and re-stitched into random partitions designated for multiple TEE-protected aggregators. As a consequence, each aggregator only sees a skewed and shuffled view of model updates and is unaware of the model architecture.

5.5 Technical Approaches and Utility Cost

In short, in comparison to present machine learning solutions, it is impossible to implement privacy-preserving machine learning without sacrificing utility, as discussed previously. In this section, we summarize and discuss existing privacy-preserving approaches in terms of their *primary technical design* and *utility cost*. The principal technical approach illustrates how these techniques fundamentally address privacy concerns, while the utility cost reflects the potential negative impact of employing these privacy-preserving techniques to achieve PPML solutions.

The fundamental technical design of existing privacy-preserving approaches is summarized in Table 2, along with their possible utility cost. The utility cost is composed of the following aspects: *model utility*, *computation utility*, *communication utility*, *scalability utility*, *scalability utility*, *scenario utility* and *privacy strength utility*. Specifically, typical anonymity strategies seek to eliminate identifiers or quasi-identifiers in order to prevent the leakage of private information, which may result in a reduction

in model utility (i.e., model accuracy). Similarly, privacy-preserving techniques that result in model utility costs include differential privacy mechanisms that inject noise into private information and approximation techniques such as sketching and compression techniques that are typically applied to intermediate model gradients. The model cryptographic privacy-preserving approaches, such as homomorphic and functional encryption, provide confidential-level privacy and enable computation over ciphertext, resulting in a loss of computation utility. Additionally, the generic 2PC or MPC offers secure computation through the use of boolean garbled circuits and oblivious transfer, which adds overhead to communication transmission. Likewise, mixed-protocol systems optimize standard multi-party computation by incorporating additional techniques such as secret sharing, arithmetic corrupted circuits, and fully homomorphic encryption, but at the expense of communication and computation utility. Emerging pairwise additive mask techniques concentrate exclusively on secure aggregation rather than generic secure computation, resulting in efficient computation due to the additive random mask protecting the private data. This system, however, requires multiple rounds of communication to agree on the random nonce and shared secret. More recently, TEEs-based techniques provide a secure compute platform but suffer from limited hardware enclave size, limiting the scalability of machine learning solutions for large model sizes. Additionally, the knowledge transfer architecture (e.g., PATE) is scenario-specific. Due to the privacy leakage caused by intermediate model gradients, the federated learning paradigm is incapable of providing a solid guarantee of privacy.

6 Challenges and Potential Directions

Although the privacy-preserving topics were not fresh in the field of data mining [115, 189], privacy-preserving machine learning remains an active and ongoing research area due to (i) the increasing adoption of traditional privacy protection mechanisms and recently proposed privacy-preserving techniques; (ii) rapidly evolving machine learning models such as emerging deep neural networks models; and (iii) the emergence of stringent privacy-related policies and legislation. Despite the fact that numerous privacy-preserving machine learning methods have been offered to meet compliance review and mitigate privacy concerns, a number of crucial challenges remain unexplored. This section highlights open issues and challenges in PPML research prior to summarizing a few interesting research directions.

Notably, the open problems and challenges in vanilla machine learning systems also apply to the PPML domain, as a PPML system is constructed on top of a vanilla machine learning system. However, this paper underscores the open problems and obstacles associated with privacy protection in the machine learning system. As a result, we suggest readers to surveys or systemization of knowledge publications, such as those in [23, 24, 25, 26], for open topics and challenges in the vanilla machine learning research area, rather than replicating the content here.

6.1 Open Problems and Challenges

The evaluation of vanilla machine learning systems is primarily concerned on model performance and system efficiency, with model performance referring to model accuracy, robustness, and fairness, and system efficiency referring to training or inference time reduction. Aside from that, existing PPML systems require additional work to ensure privacy. Designing a well-designed PPML solution entails addressing the following open issues:

(i) In terms of privacy protection, how can a PPML solution be assured of adequate privacy protection in accordance with the trust assumption and threat model settings? Generally, the privacy guarantee should be as robust as possible from the data owners' standpoint.

(ii) In terms of model accuracy, how can we ensure that the trained model in the PPML approach is as accurate as the model trained in the contrasted vanilla machine learning system without using any privacy-preserving settings?

(iii) In terms of model robustness and fairness, how can we add privacy-preserving capabilities without impairing the model's robustness and fairness?

(iv) In terms of system performance, how can the PPML system communicate and compute as effectively as the vanilla machine learning system?

As the premise of a well-known adage - there is no such thing as a free lunch - implies, it is not free to power a vanilla machine learning system with privacy-protecting capabilities while keeping

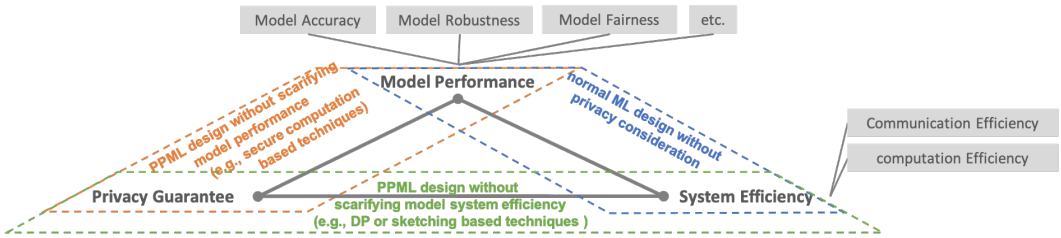

Figure 4: An illustration of the trade-offs that will be made when creating an optimal PPML solution among privacy assurance, model performance, and system efficiency.

other system qualities such as model performance and efficiency. In general, those open problems are mutually incompatible. For example, in the vanilla deep neural networks model, increased model accuracy indicates the presence of numerous layers of neural networks, which requires additional training data and training epochs to achieve coverage. As explained in Section 5, using differential privacy as an example, implementing privacy-preserving techniques reduces model accuracy to a certain extent. Specifically, Abadi et al. [36] propose injecting the DP budget(noise) into the DP-SGD based deep learning model training; however, their evaluation results indicate that the model's accuracy cannot match that of the originally trained deep learning model. Bagdasaryan et al. [190] recently demonstrated a compatibility issue between privacy and fairness, showing and explaining that if the original model is unfair, the unfairness is exacerbated when the DP technique is used.

In summary, we believe that the primary challenge in developing an ideal PPML solution is resolving a trade-off dilemma between the solutions to those open problems, as illustrated in Figure 4. Existing PPML solutions trade either *system efficiency* or *model performance* for *privacy guarantee*. The efficiency of vanilla machine learning systems is concerned with how to improve training or inference efficiency, particularly for emerging deep neural network models with complicated network structures. The majority of the machine learning community's efforts have been directed either increasing computation power, training in a distributed manner, or parallel training in order to address these efficiency issues. Additionally, as indicated in Figure 4, the efficiency issue with PPML systems highlights the following two concerns:

(i) PPML's communication efficiency emphasizes the importance of securely computing a function with fewer communication interactions and low transmission overhead;

(ii) PPML's computation efficiency emphasizes the need of securely computing a function with a low computational overhead or an acceptable computation time in the context of complicated machine learning training.

Besides that, as discussed and summarized above, the majority of existing PPML approaches focus on providing some degree of privacy-preserving features to a specific machine learning system; however, there is still a dearth of systematic definitions of privacy guarantees in terms of threat models or trust settings. That is the reason that we present our initial summary on the privacy guarantee notations in Section 4. We expect that the presentation of privacy guarantees and the corresponding trust assumption and threat model settings will shed light on future PPML research. Indeed, achieving broad agreement on a privacy guarantee definition for PPML systems remains a challenge.

6.2 Research Directions

This section summarizes our insights on the future directions of research in cross-domain of privacy-preserving techniques and machine learning.

6.2.1 Systematic Definition, Measurement and Evaluation of Privacy

As discussed in Sections 3 and 5, various PPML solutions have demonstrated their efforts to provide machine learning systems with privacy guarantees and to include formal or informal privacy analysis to prove the privacy guarantees they claimed. However, neither those PPML solutions nor a set of criteria could give a framework for evaluating or classifying the degree of privacy guarantee.

While we have provided a descriptive measurement and summary of the privacy guarantee in Section 4 in terms of the trust assumption and threat model settings for each entity in the machine learning pipeline, we believe it is insufficient for the PPML investigation. There is still a need for a formal and systematic definition of the PPML system's privacy guarantee, as well as a commonly accepted framework or approach for evaluating the degree of privacy protection that a PPML system can provide.

6.2.2 Attack and Defense Strategies

In general, the machine learning system is vulnerable to three types of attacks: (i) poisoning attacks that compromise the integrity of the training dataset collection; (ii) inference attacks that infer private information from an individual participant's training data (or intermediate model update) or the trained (or aggregated) model; and (iii) evasion/exploratory attacks that cause the trained model to produce incorrect (targeted/untargeted) classification outputs or collect evidence.

In terms of poisoning attack, several typical attacks include, for instance, clean-label data poisoning attack [191] where the adversary is assumed not to change the label of any training data and hence the poisoning of data samples has to be imperceptible, dirty-label data poisoning attack [192] where the adversary is assumed to introduce a number of data sample with miss-classified and desired target label into the training set, and model poisoning [193] where the adversary can poison local model updates before sending them to the server or insert hidden backdoors into the global model.

Regarding the inference attack, typical examples include membership inference attacks [11, 12, 13, 14, 15], in which an attacker can infer whether a specific patient profile was used to train a classifier associated with a disease; model inversion attacks [16, 18, 181, 17] that can use black-box access to prediction models in order to estimate aspects of someone's genomics information; deep leakage from gradients [182, 22, 183] that obtains the private training data from the shared gradients in the ML cases of computer vision and natural language processing tasks.

PPML research aims to prevent the leakage of private information, therefore acting as a defender to such attacks. As the adage goes, "know yourself and your enemy, and you will never be defeated.", understanding the fundamentals of those attacks will guide the research group to incorporate appropriate privacy-preserving strategies into the PPML solutions. Additionally, it is worthwhile to investigate novel privacy-preserving approaches or to enhance existing methods in light of newly demonstrated inference attacks.

6.2.3 Communication Efficiency

As discussed in Section 5, techniques such as boolean or arithmetic garbled-circuits, secret sharing, and oblivious transfer are widely used to construct a secure generic multi-party computation approach, which has been recently used in PPML solutions to provide privacy protections for the participant's input. Based on the decomposition of the garbled circuits, it can be observed that each input should be encoded with associated keys and form different types of permuted garbled tables. As a result, this approach places a significant burden on protocol communication in terms of training and inference; this is especially true when dealing with recent deep neural network models that involve complex network architecture and rely on massive amounts of training data, such as the most recent GPT-3 language model that was trained over 45TB of data and contains 175 billion parameters [194]. Additionally, the new approach to secure aggregation based on pairwise masking and secret sharing relies on multi-round peer-to-peer communication.

Two directions could be considered here to increase the communication efficiency of secure multi-party computation. The first direction is to optimize the PPML solution's computation procedures. For example, we may want to minimize the computational complexity of traditional machine learning systems or to improve the architecture of deep neural networks without impairing model performance. The second option may involve optimizing garbled-circuits MPC protocols, for example, by offering a well-designed compiler that generates fewer garbled-circuits gates and thus reduces data transmission size. It is worthwhile to optimize the secret-sharing strategy or communication topology when it comes to pairwise masking-based secure aggregation.

6.2.4 Computation Efficiency

Similarly to the optimization of garbled-circuits-based secure computation approaches, the optimization of emerging crypto-based secure computation approaches can also focus on optimizing the machine learning model to reduce the computing burden.

Another alternative is to develop an efficient cryptography technique that enables computation over ciphertext, such as homomorphic or functional encryption schemes. Specifically, despite the cryptographic community's efforts to propose various types of homomorphic- or functional-encryption schemes with the goal of providing formal security proofs, increasing security degrees, or supporting more generic functionality, there is still a dearth of efforts to construct simple, practical, functionality-specific encryption schemes. We recommend the reader to [158] for a comprehensive study of the challenges and directions for secure computation using emerging and promising functional encryption approaches. Particularly in the context of edge computing, where IoT sensors or mobile devices have low computational capabilities, such a demand emerges in the crypto-based PPML system.

Furthermore, with the exception of homomorphic encryption systems based on approximate number arithmetic, such as CKKS [77], the majority of secure computation-related cryptography schemes perform on the integer group. Thus, determining the encoding precision is another problem to resolve, as noted above, because the majority of crypto-based PPML systems rely on the conversion of integer and floating-point numbers between the cryptosystem and the machine learning system. In general, less precise encoding implies more efficient secure computation, which results in poorer model accuracy, and vice versa.

6.2.5 Privacy Perturbation Budget and Model Utility

As previously stated, it is impossible to incorporate a privacy perturbation budget into a machine learning system without impairing model utility. A prominent example is the recently popular privacy-preserving technique, (ϵ, δ)-differential privacy mechanism, in which ϵ denotes the privacy budget. As proved in existing proposals such as [36, 195], the privacy budget ϵ negatively correlates to the model accuracy in the DP-based PPML solution. To be more precise, if the PPML system chases a higher privacy budget (a.k.a., a tighter privacy guarantee), it will diminish model accuracy; however, a lower privacy budget implies a greater likelihood of privacy inference attacks succeeding.

As a result, two prospective avenues for research are as follows: (i) How can we determine an appropriate privacy budget in light of the likelihood of inference attacks and model accuracy, or are there any universal or dynamic approaches for determining the privacy budget for various machine learning models? (ii) How to minimize the local privacy budget for each participant in a distributed training environment without compromising the final global model's privacy budget?

6.2.6 New Deployment Approaches of Differential Privacy in PPML

The emerging adoption of differential privacy mechanisms in PPML focuses on two directions: (i) *centralized differential privacy (CDP)* deployment approach in which the DP noise is injected by a centralized trust node that has access to all private data, and (ii) *local differential privacy (LDP)* adoption where each individual perturbs private data or corresponding ML model before sending out in distributed ML scenarios. An example of CDP adoption is the differentially private stochastic gradient descent (DP-SGD) based deep neural networks training [190]. Examples of the LDP method being used in PPML include [52, 35, 57]. LDP's enhanced privacy qualities come at the expense of model utility. As a result, it is worthwhile to investigate the new deployment of DP approaches in PPML solutions in order to ensure adequate privacy guarantees while increasing model utility via cryptographic primitives, anonymous communication techniques, or a newly designed machine learning training architecture.

6.2.7 Compatibility of Privacy, Fairness, and Robustness

Recent proposals [190, 196] demonstrate that using DP-SGD in neural network training has a disparate effect on model accuracy; specifically, accuracy reduces significantly more for underrepresented classes and subgroups, implying that DP-SGD exacerbates the model fairness issue. In particular, it is impossible to establish differential privacy and perfect fairness while keeping non-trivial accuracy, even when we have access to the entire distribution of the data. Regarding numerous other related methods, such as sketching algorithms, there is still a dearth of research into the feasibility of

achieving both privacy and fairness concurrently. Additionally, we must demonstrate why the differential privacy and fairness are incompatible. Understanding the cause for compatibility may help in mitigating the DP mechanism's impact on the model's fairness.

On the other side, the relevance of privacy and model robustness still lacks sufficient exploring and study. Few recent studies, such as those in [197, 198, 199], have explored the impact of the DP mechanism on poisoning attacks against deep neural networks and backdoor attacks against federated learning, namely, to what extent DP can be used to protect not only privacy but also robustness in ML. In short, it seems like both LDP and CDP can provide a substantially more vigorous defense against backdoor and poisoning attacks in practice than in theory. There is a dearth of research into the utility of alternative privacy-preserving approaches and their impact on the robustness of generic models.

6.2.8 Novel Architecture of PPML

Existing architecture-based PPML solutions, such as the federated learning paradigm or the PATE framework, have demonstrated promising results in terms of privately training models over independent datasets while resolving data silo issues. However, there are still challenges to be addressed in the FL systems, such as inefficient communication, systems, statistical heterogeneity, and potential privacy leakages. The reader is referred to [25, 26, 23, 27, 28] for a comprehensive review of the FL systems' remaining open problems. Additionally, in addition to existing FL [4, 5] and PATE [78, 79] paradigms, it is worthwhile to investigate novel distributed machine learning architectures for privacy preservation.

6.2.9 New Model Publishing Method for PPML

Differential privacy (DP) is the most widely used mechanism for publishing a (locally) trained machine learning model in the context of the DP-SGD training method, model aggregation in the FL system, or the machine learning as a service (MLaaS) architecture. As noted previously, the DP's limitations are visible in the trade-offs between privacy budget and model accuracy, as well as the recently proposed compatibility issue. Recent proposals to incorporate classic sketching techniques such as the count-min sketch - a time-honored approach for measuring network traffic or approximation query processing - into the FL system have demonstrated their promise in terms of privacy protection [98, 43]. It is still worthwhile to investigate different sorts of sketching techniques, as well as their application breadth and restrictions. Apart from the DP and sketching techniques discussed previously, are there any other approximate methods that might be considered candidates for the privacy-preserving model publishing method?

6.2.10 Interpretability in PPML

Researchers in the ML domain have recently started to put efforts into the interpretability of deep learning models since the deep neural networks model is still not explainable to its results. The interpretability study allows us to know the deep learning model, and hence it is also helpful to give us insights into how private information is disclosed. As a result, the interpretability study may help employ proper privacy-preserving approaches or design new privacy-preserving methods. Additionally, the PPML system itself lacks explanation, such as the interpretability of the impact and the effects of applying the privacy-preserving approach.

6.2.11 Benchmarking

Benchmarking, in the context of PPML, is the practice of comparing tools in order to identify the most performant PPML solutions, which does not receive adequate investigation. It is worthwhile to construct a privacy-preserving benchmarking framework similar to LEAF [200], a framework for comparing popular learning tasks, methods, and datasets in federated environments. To evaluate PPML's performance in terms of privacy protection, model accuracy, fairness, and robustness, the framework could incorporate synthetic private datasets, privacy-preserving methods, inference attack toolkits such as the Adversarial Robustness Toolbox (ART)[8], and common machine learning models.

[8] https://github.com/Trusted-AI/adversarial-robustness-toolbox

7 Conclusion

In this paper, we summarize and discuss existing privacy-preserving methodologies used in machine learning systems from a variety of perspectives, including the phases of the machine learning system and the underlying design principles, as well as provide the corresponding most recent PPML proposals. Additionally, we evaluate the privacy guarantee by defining several levels of privacy guarantee in the PPML systems. Finally, we have outlined the challenges and open problems as well as pointing out future directions. Solving those problems will require interdisciplinary effort from machine learning, distributed systems, and security-privacy communities.

References

[1] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[2] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[3] Iyad Rahwan, Manuel Cebrian, Nick Obradovich, Josh Bongard, Jean-François Bonnefon, Cynthia Breazeal, Jacob W Crandall, Nicholas A Christakis, Iain D Couzin, Matthew O Jackson, et al. Machine behaviour. *Nature*, 568(7753):477–486, 2019.

[4] H Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, et al. Communication-efficient learning of deep networks from decentralized data. *arXiv preprint arXiv:1602.05629*, 2016.

[5] Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

[6] Jie Xu and Fei Wang. Federated learning for healthcare informatics. *arXiv preprint arXiv:1911.06270*, 2019.

[7] Sumudu Samarakoon, Mehdi Bennis, Walid Saad, and Mérouane Debbah. Distributed federated learning for ultra-reliable low-latency vehicular communications. *IEEE Transactions on Communications*, 68(2):1146–1159, 2019.

[8] Meng Hao, Hongwei Li, Xizhao Luo, Guowen Xu, Haomiao Yang, and Sen Liu. Efficient and privacy-enhanced federated learning for industrial artificial intelligence. *IEEE Transactions on Industrial Informatics*, 2019.

[9] Pierangelo Rosati, Peter Deeney, Mark Cummins, Lisa Van der Werff, and Theo Lynn. Social media and stock price reaction to data breach announcements: Evidence from us listed companies. *Research in International Business and Finance*, 47:458–469, 2019.

[10] Naga Vemprala and Glenn Dietrich. A social network analysis (sna) study on data breach concerns over social media. In *Proceedings of the 52nd Hawaii International Conference on System Sciences*, 2019.

[11] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 3–18. IEEE, 2017.

[12] Daniel Bernau, Philip-William Grassal, Jonas Robl, and Florian Kerschbaum. Assessing differentially private deep learning with membership inference. *arXiv preprint arXiv:1912.11328*, 2019.

[13] Jinyuan Jia, Ahmed Salem, Michael Backes, Yang Zhang, and Neil Zhenqiang Gong. Memguard: Defending against black-box membership inference attacks via adversarial examples. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 259–274, 2019.

[14] Jiacheng Li, Ninghui Li, and Bruno Ribeiro. Membership inference attacks and defenses in supervised learning via generalization gap. *arXiv preprint arXiv:2002.12062*, 2020.

[15] Milad Nasr, Reza Shokri, and Amir Houmansadr. Machine learning with membership privacy using adversarial regularization. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 634–646, 2018.

[16] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 1322–1333, 2015.

[17] Zecheng He, Tianwei Zhang, and Ruby B Lee. Model inversion attacks against collaborative inference. In *Proceedings of the 35th Annual Computer Security Applications Conference*, pages 148–162, 2019.

[18] Xi Wu, Matthew Fredrikson, Somesh Jha, and Jeffrey F Naughton. A methodology for formalizing model-inversion attacks. In *2016 IEEE 29th Computer Security Foundations Symposium (CSF)*, pages 355–370. IEEE, 2016.

[19] Karan Ganju, Qi Wang, Wei Yang, Carl A Gunter, and Nikita Borisov. Property inference attacks on fully connected neural networks using permutation invariant representations. In *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*, pages 619–633, 2018.

[20] Mathias PM Parisot, Balazs Pejo, and Dayana Spagnuelo. Property inference attacks on convolutional neural networks: Influence and implications of target model's complexity. *arXiv preprint arXiv:2104.13061*, 2021.

[21] Ligeng Zhu, Zhijian Liu, and Song Han. Deep leakage from gradients. *Advances in Neural Information Processing Systems*, 32:14774–14784, 2019.

[22] Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. idlg: Improved deep leakage from gradients. *arXiv preprint arXiv:2001.02610*, 2020.

[23] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Keith Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*, 2019.

[24] Samira Pouyanfar, Saad Sadiq, Yilin Yan, Haiman Tian, Yudong Tao, Maria Presa Reyes, Mei-Ling Shyu, Shu-Ching Chen, and SS Iyengar. A survey on deep learning: Algorithms, techniques, and applications. *ACM Computing Surveys (CSUR)*, 51(5):1–36, 2018.

[25] Qinbin Li, Zeyi Wen, and Bingsheng He. Federated learning systems: Vision, hype and reality for data privacy and protection. *arXiv preprint arXiv:1907.09693*, 2019.

[26] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2):1–19, 2019.

[27] Lingjuan Lyu, Han Yu, and Qiang Yang. Threats to federated learning: A survey. *arXiv preprint arXiv:2003.02133*, 2020.

[28] Xuefei Yin, Yanming Zhu, and Jiankun Hu. A comprehensive survey of privacy-preserving federated learning: A taxonomy, review, and future directions. *ACM Computing Surveys (CSUR)*, 54(6):1–36, 2021.

[29] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 169–178, 2009.

[30] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In *Theory of Cryptography Conference*, pages 253–273. Springer, 2011.

[31] Nicolas Papernot, Patrick McDaniel, Arunesh Sinha, and Michael P Wellman. Sok: Security and privacy in machine learning. In *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 399–414. IEEE, 2018.

[32] Marco Barreno, Blaine Nelson, Russell Sears, Anthony D Joseph, and J Doug Tygar. Can machine learning be secure? In *Proceedings of the 2006 ACM Symposium on Information, computer and communications security*, pages 16–25, 2006.

[33] Marco Barreno, Blaine Nelson, Anthony D Joseph, and J Doug Tygar. The security of machine learning. *Machine Learning*, 81(2):121–148, 2010.

[34] Milad Nasr, Reza Shokri, and Amir Houmansadr. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In *2019 IEEE symposium on security and privacy (SP)*, pages 739–753. IEEE, 2019.

[35] Runhua Xu, Nathalie Baracaldo, Yi Zhou, Ali Anwar, and Heiko Ludwig. Hybridalpha: An efficient approach for privacy-preserving federated learning. In *Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security*, pages 13–23, 2019.

[36] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 308–318. ACM, 2016.

[37] Runhua Xu, James Joshi, and Chao Li. Nn-emd: Efficiently training neural networks using encrypted multi-sourced datasets. *IEEE Transactions on Dependable and Secure Computing*, 2021.

[38] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1175–1191. ACM, 2017.

[39] Latanya Sweeney. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(05):557–570, 2002.

[40] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1):3–es, 2007.

[41] Ninghui Li, Tiancheng Li, and Suresh Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and l-diversity. In *2007 IEEE 23rd International Conference on Data Engineering*, pages 106–115. IEEE, 2007.

[42] Mengwei Yang, Linqi Song, Jie Xu, Congduan Li, and Guozhen Tan. The tradeoff between privacy and accuracy in anomaly detection using federated xgboost. *arXiv preprint arXiv:1907.07157*, 2019.

[43] Tian Li, Zaoxing Liu, Vyas Sekar, and Virginia Smith. Privacy for free: Communication-efficient learning with differential privacy using sketches. *arXiv preprint arXiv:1911.00972*, 2019.

[44] Farzin Haddadpour, Belhal Karimi, Ping Li, and Xiaoyun Li. Fedsketch: Communication-efficient and private federated learning via sketching. *arXiv preprint arXiv:2008.04975*, 2020.

[45] Cynthia Dwork. Differential privacy: A survey of results. In *International conference on theory and applications of models of computation*, pages 1–19. Springer, 2008.

[46] Cynthia Dwork, Guy N Rothblum, and Salil Vadhan. Boosting and differential privacy. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pages 51–60. IEEE, 2010.

[47] Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014.

[48] Arik Friedman, Ran Wolff, and Assaf Schuster. Providing k-anonymity in data mining. *The VLDB Journal*, 17(4):789–804, 2008.

[49] Kristen LeFevre, David J DeWitt, and Raghu Ramakrishnan. Workload-aware anonymization. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 277–286, 2006.

[50] Daniel Kifer and Johannes Gehrke. Injecting utility into anonymized datasets. In *Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, pages 217–228, 2006.

[51] H Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. Learning differentially private recurrent language models. *arXiv preprint arXiv:1710.06963*, 2017.

[52] Robin C Geyer, Tassilo Klein, and Moin Nabi. Differentially private federated learning: A client level perspective. *arXiv preprint arXiv:1712.07557*, 2017.

[53] Jeffrey Li, Mikhail Khodak, Sebastian Caldas, and Ameet Talwalkar. Differentially private meta-learning. *arXiv preprint arXiv:1909.05830*, 2019.

[54] Gilbert Wondracek, Thorsten Holz, Engin Kirda, and Christopher Kruegel. A practical attack to de-anonymize social network users. In *2010 IEEE Symposium on Security and Privacy*, pages 223–238. IEEE, 2010.

[55] Md Atiqur Rahman, Tanzila Rahman, Robert Laganière, Noman Mohammed, and Yang Wang. Membership inference attack against differentially private deep learning model. *Transactions on Data Privacy*, 11(1):61–79, 2018.

[56] Jianwei Qian, Xiang-Yang Li, Chunhong Zhang, and Linlin Chen. De-anonymizing social networks and inferring private attributes using knowledge graphs. In *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*, pages 1–9. IEEE, 2016.

[57] Stacey Truex, Nathalie Baracaldo, Ali Anwar, Thomas Steinke, Heiko Ludwig, Rui Zhang, and Yi Zhou. A hybrid approach to privacy-preserving federated learning. In *Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security*, pages 1–11, 2019.

[58] Runhua Xu, James Joshi, and Chao Li. Cryptonn:training neural networks over encrypted data. In *2019 39th IEEE International Conference on Distributed Computing Systems (ICDCS)*, pages 1199–1209. IEEE, 2019.

[59] Adrià Gascón, Phillipp Schoppmann, Borja Balle, Mariana Raykova, Jack Doerner, Samee Zahur, and David Evans. Secure linear regression on vertically partitioned datasets. *IACR Cryptology ePrint Archive*, 2016:892, 2016.

[60] Stephen Hardy, Wilko Henecka, Hamish Ivey-Law, Richard Nock, Giorgio Patrini, Guillaume Smith, and Brian Thorne. Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption. *arXiv preprint arXiv:1711.10677*, 2017.

[61] Kewei Cheng, Tao Fan, Yilun Jin, Yang Liu, Tianjian Chen, and Qiang Yang. Secureboost: A lossless federated learning framework. *arXiv preprint arXiv:1901.08755*, 2019.

[62] Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *International Conference on Machine Learning*, pages 201–210, 2016.

[63] Hervé Chabanne, Amaury de Wargny, Jonathan Milgram, Constance Morel, and Emmanuel Prouff. Privacy-preserving classification on deep neural network. *IACR Cryptology ePrint Archive*, 2017:35, 2017.

[64] Karthik Nandakumar, Nalini Ratha, Sharath Pankanti, and Shai Halevi. Towards deep neural network training on encrypted data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019.

[65] Marten Van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 24–43. Springer, 2010.

[66] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. *ACM Transactions on Computation Theory (TOCT)*, 6(3):1–36, 2014.

[67] Paulo Martins, Leonel Sousa, and Artur Mariano. A survey on fully homomorphic encryption: An engineering perspective. *ACM Computing Surveys (CSUR)*, 50(6):1–33, 2017.

[68] Abbas Acar, Hidayet Aksu, A Selcuk Uluagac, and Mauro Conti. A survey on homomorphic encryption schemes: Theory and implementation. *ACM Computing Surveys (CSUR)*, 51(4):1–35, 2018.

[69] Shafi Goldwasser, S Dov Gordon, Vipul Goyal, Abhishek Jain, Jonathan Katz, Feng-Hao Liu, Amit Sahai, Elaine Shi, and Hong-Sheng Zhou. Multi-input functional encryption. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 578–602. Springer, 2014.

[70] Michel Abdalla, Florian Bourse, Angelo De Caro, and David Pointcheval. Simple functional encryption schemes for inner products. In *IACR International Workshop on Public Key Cryptography*, pages 733–751. Springer, 2015.

[71] Nuttapong Attrapadung and Benoît Libert. Functional encryption for inner product: Achieving constant-size ciphertexts with adaptive security or support for negation. In *International Workshop on Public Key Cryptography*, pages 384–402. Springer, 2010.

[72] Prabhanjan Ananth and Vinod Vaikuntanathan. Optimal bounded-collusion secure functional encryption. In *Theory of Cryptography Conference*, pages 174–198. Springer, 2019.

[73] Michel Abdalla, Fabrice Benhamouda, and Romain Gay. From single-input to multi-client inner-product functional encryption. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 552–582. Springer, 2019.

[74] Michel Abdalla, Fabrice Benhamouda, Markulf Kohlweiss, and Hendrik Waldner. Decentralizing inner-product functional encryption. In *IACR International Workshop on Public Key Cryptography*, pages 128–157. Springer, 2019.

[75] Michel Abdalla, Dario Catalano, Dario Fiore, Romain Gay, and Bogdan Ursu. Multi-input functional encryption for inner products: function-hiding realizations and constructions without pairings. In *Annual International Cryptology Conference*, pages 597–627. Springer, 2018.

[76] Masahiro Yagisawa. Fully homomorphic encryption without bootstrapping. *IACR Cryptol. EPrint Arch.*, 2015:474, 2015.

[77] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Homomorphic encryption for arithmetic of approximate numbers. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 409–437. Springer, 2017.

[78] Nicolas Papernot, Martín Abadi, Ulfar Erlingsson, Ian Goodfellow, and Kunal Talwar. Semi-supervised knowledge transfer for deep learning from private training data. *arXiv preprint arXiv:1610.05755*, 2016.

[79] Nicolas Papernot, Shuang Song, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Úlfar Erlingsson. Scalable private learning with pate. In *Proceedings of the 2018 Sixth International Conference on Learning Representations*, 2018.

[80] Chong Liu, Yuqing Zhu, Kamalika Chaudhuri, and Yu-Xiang Wang. Revisiting model-agnostic private learning: Faster rates and active learning. *arXiv preprint arXiv:2011.03186*, 2020.

[81] Jian Liu, Mika Juuti, Yao Lu, and Nadarajah Asokan. Oblivious neural network predictions via minionn transformations. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 619–631, 2017.

[82] Deevashwer Rathee, Mayank Rathee, Rahul Kranti Kiran Goli, Divya Gupta, Rahul Sharma, Nishanth Chandran, and Aseem Rastogi. Sirnn: A math library for secure rnn inference. *arXiv preprint arXiv:2105.04236*, 2021.

[83] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[84] Antonio Polino, Razvan Pascanu, and Dan Alistarh. Model compression via distillation and quantization. *arXiv preprint arXiv:1802.05668*, 2018.

[85] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 582–597. IEEE, 2016.

[86] Ji Wang, Weidong Bao, Lichao Sun, Xiaomin Zhu, Bokai Cao, and S Yu Philip. Private model compression via knowledge distillation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 1190–1197, 2019.

[87] Payman Mohassel and Yupeng Zhang. Secureml: A system for scalable privacy-preserving machine learning. In *2017 38th IEEE Symposium on Security and Privacy (SP)*, pages 19–38. IEEE, 2017.

[88] Bita Darvish Rouhani, M Sadegh Riazi, and Farinaz Koushanfar. Deepsecure: Scalable provably-secure deep learning. In *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2018.

[89] Azalia Mirhoseini, Ahmad-Reza Sadeghi, and Farinaz Koushanfar. Cryptoml: Secure outsourcing of big data machine learning applications. In *Hardware Oriented Security and Trust (HOST), 2016 IEEE International Symposium on*, pages 149–154. IEEE, 2016.

[90] Daniel J Solove. *Understanding privacy*. Harvard University Press, May, 2008.

[91] Cynthia Dwork and Jing Lei. Differential privacy and robust statistics. In *STOC*, volume 9, pages 371–380. ACM, 2009.

[92] Yuya Jeremy Ong, Yi Zhou, Nathalie Baracaldo, and Heiko Ludwig. Adaptive histogram-based gradient boosted trees for federated learning. *arXiv preprint arXiv:2012.06670*, 2020.

[93] Ilya Mironov. Rényi differential privacy. In *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*, pages 263–275. IEEE, 2017.

[94] Rui Chen, Noman Mohammed, Benjamin CM Fung, Bipin C Desai, and Li Xiong. Publishing set-valued data via differential privacy. *Proceedings of the VLDB Endowment*, 4(11):1087–1098, 2011.

[95] Kaifeng Jiang, Dongxu Shao, Stéphane Bressan, Thomas Kister, and Kian-Lee Tan. Publishing trajectories with differential privacy guarantees. In *Proceedings of the 25th International Conference on Scientific and Statistical Database Management*, pages 1–12, 2013.

[96] Charu C Aggarwal and Philip S Yu. On privacy-preservation of text and sparse binary data with sketches. In *Proceedings of the 2007 SIAM International Conference on Data Mining*, pages 57–67. SIAM, 2007.

[97] Luca Melis, George Danezis, and Emiliano De Cristofaro. Efficient private statistics with succinct sketches. *arXiv preprint arXiv:1508.06110*, 2015.

[98] Raghavendran Balu and Teddy Furon. Differentially private matrix factorization using sketching techniques. In *Proceedings of the 4th ACM Workshop on Information Hiding and Multimedia Security*, pages 57–62, 2016.

[99] James Jordon, Jinsung Yoon, and Mihaela van der Schaar. Pate-gan: Generating synthetic data with differential privacy guarantees. In *The International Conference on Learning Representations (ICLR)*, 2019.

[100] Aleksei Triastcyn and Boi Faltings. Generating artificial data for private deep learning. *arXiv preprint arXiv:1803.03148*, 2018.

[101] Liyang Xie, Kaixiang Lin, Shu Wang, Fei Wang, and Jiayu Zhou. Differentially private generative adversarial network. *arXiv preprint arXiv:1802.06739*, 2018.

[102] Liyue Fan. A survey of differentially private generative adversarial networks. In *The AAAI Workshop on Privacy-Preserving Artificial Intelligence*, 2020.

[103] Payman Mohassel, Mike Rosulek, and Ye Zhang. Fast and secure three-party computation: The garbled circuit approach. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 591–602, 2015.

[104] Xiao Wang, Samuel Ranellucci, and Jonathan Katz. Global-scale secure multiparty computation. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 39–56, 2017.

[105] Benny Pinkas, Thomas Schneider, Nigel P Smart, and Stephen C Williams. Secure two-party computation is practical. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 250–267. Springer, 2009.

[106] M Sadegh Riazi, Christian Weinert, Oleksandr Tkachenko, Ebrahim M Songhori, Thomas Schneider, and Farinaz Koushanfar. Chameleon: A hybrid secure computation framework for machine learning applications. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, pages 707–721, 2018.

[107] Andrew C Yao. Protocols for secure computations. In *23rd annual symposium on foundations of computer science (sfcs 1982)*, pages 160–164. IEEE, 1982.

[108] Ronald Cramer, Ivan Bjerre Damgård, and Jesper Buus Nielsen. *Secure multiparty computation*. Cambridge University Press, 2015.

[109] Marcella Hastings, Brett Hemenway, Daniel Noble, and Steve Zdancewic. Sok: General purpose compilers for secure multi-party computation. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 1220–1237. IEEE, 2019.

[110] Ashish P Sanil, Alan F Karr, Xiaodong Lin, and Jerome P Reiter. Privacy preserving regression modelling via distributed computation. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 677–682, 2004.

[111] Swanand Kadhe, Nived Rajaraman, O Ozan Koyluoglu, and Kannan Ramchandran. Fastsecagg: Scalable secure aggregation for privacy-preserving federated learning. *arXiv preprint arXiv:2009.11248*, 2020.

[112] Jinhyun So, Başak Güler, and A Salman Avestimehr. Turbo-aggregate: Breaking the quadratic aggregation barrier in secure federated learning. *IEEE Journal on Selected Areas in Information Theory*, 2021.

[113] David Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *J. Cryptology*, 1(1):65–75, 1988.

[114] David L Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–90, 1981.

[115] Rakesh Agrawal and Ramakrishnan Srikant. Privacy-preserving data mining. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pages 439–450, 2000.

[116] Hillol Kargupta, Souptik Datta, Qi Wang, and Krishnamoorthy Sivakumar. Random-data perturbation techniques and privacy-preserving data mining. *Knowledge and Information Systems*, 7(4):387–414, 2005.

[117] Paul Bunn and Rafail Ostrovsky. Secure two-party k-means clustering. In *Proceedings of the 14th ACM conference on Computer and communications security*, pages 486–497, 2007.

[118] Mahir Can Doganay, Thomas B Pedersen, Yücel Saygin, Erkay Savaş, and Albert Levi. Distributed privacy preserving k-means clustering with additive secret sharing. In *Proceedings of the 2008 international workshop on Privacy and anonymity in information society*, pages 3–11, 2008.

[119] Aner Ben-Efraim, Yehuda Lindell, and Eran Omri. Optimizing semi-honest secure multiparty computation for the internet. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 578–590, 2016.

[120] Xiao Wang, Samuel Ranellucci, and Jonathan Katz. Authenticated garbling and efficient maliciously secure two-party computation. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 21–37, 2017.

[121] Jonathan Katz, Samuel Ranellucci, Mike Rosulek, and Xiao Wang. Optimizing authenticated garbling for faster secure two-party computation. In *Annual International Cryptology Conference*, pages 365–391. Springer, 2018.

[122] Nishanth Chandran, Divya Gupta, Aseem Rastogi, Rahul Sharma, and Shardul Tripathi. Ezpc: Programmable and efficient secure two-party computation for machine learning. In *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 496–511. IEEE, 2019.

[123] Payman Mohassel and Peter Rindal. Aby3: A mixed protocol framework for machine learning. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 35–52, 2018.

[124] Daniel Demmler, Thomas Schneider, and Michael Zohner. Aby-a framework for efficient mixed-protocol secure two-party computation. In *NDSS*, 2015.

[125] M Sadegh Riazi, Mohammad Samragh, Hao Chen, Kim Laine, Kristin Lauter, and Farinaz Koushanfar. {XONN}: Xnor-based oblivious deep neural network inference. In *28th {USENIX} Security Symposium ({USENIX} Security 19)*, pages 1501–1518, 2019.

[126] Nitin Agrawal, Ali Shahin Shamsabadi, Matt J Kusner, and Adrià Gascón. Quotient: two-party secure neural network training and prediction. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 1231–1247, 2019.

[127] Rob Hall, Stephen E Fienberg, and Yuval Nardi. Secure multiple linear regression based on homomorphic encryption. *Journal of Official Statistics*, 27(4):669, 2011.

[128] Valeria Nikolaenko, Udi Weinsberg, Stratis Ioannidis, Marc Joye, Dan Boneh, and Nina Taft. Privacy-preserving ridge regression on hundreds of millions of records. In *2013 IEEE Symposium on Security and Privacy*, pages 334–348. IEEE, 2013.

[129] Martine de Cock, Rafael Dowsley, Anderson CA Nascimento, and Stacey C Newman. Fast, privacy preserving linear regression over distributed datasets based on pre-distributed data. In *Proceedings of the 8th ACM Workshop on Artificial Intelligence and Security*, pages 3–14, 2015.

[130] Qian Lou, Bo Feng, Geoffrey C Fox, and Lei Jiang. Glyph: Fast and accurately training deep neural networks on encrypted data. *arXiv preprint arXiv:1911.07101*, 2019.

[131] Martin Albrecht, Melissa Chase, Hao Chen, Jintai Ding, Shafi Goldwasser, Sergey Gorbunov, Shai Halevi, Jeffrey Hoffstein, Kim Laine, Kristin Lauter, Satya Lokam, Daniele Micciancio, Dustin Moody, Travis Morrison, Amit Sahai, and Vinod Vaikuntanathan. Homomorphic encryption security standard. Technical report, HomomorphicEncryption.org, Toronto, Canada, November 2018.

[132] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *International conference on the theory and applications of cryptographic techniques*, pages 223–238. Springer, 1999.

[133] Shai Halevi and Victor Shoup. Algorithms in helib. In *Annual Cryptology Conference*, pages 554–571. Springer, 2014.

[134] Nigel P Smart and Frederik Vercauteren. Fully homomorphic simd operations. *Designs, codes and cryptography*, 71(1):57–81, 2014.

[135] Craig Gentry, Shai Halevi, and Nigel P Smart. Homomorphic evaluation of the aes circuit. In *Annual Cryptology Conference*, pages 850–867. Springer, 2012.

[136] Redmond Microsoft Research. Microsoft SEAL (release 3.5), 2020.

[137] Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 1219–1234, 2012.

[138] Diego Chialva and Ann Dooms. Conditionals in homomorphic encryption and machine learning applications. *arXiv preprint arXiv:1810.12380*, 2018.

[139] Pratyush Mishra, Ryan Lehmkuhl, Akshayaram Srinivasan, Wenting Zheng, and Raluca Ada Popa. Delphi: A cryptographic inference service for neural networks. In *29th {USENIX} Security Symposium ({USENIX} Security 20)*, pages 2505–2522, 2020.

[140] Ryan Lehmkuhl, Pratyush Mishra, Akshayaram Srinivasan, and Raluca Ada Popa. Muse: Secure inference resilient to malicious clients. In *30th {USENIX} Security Symposium ({USENIX} Security 21)*, 2021.

[141] Wenting Zheng, Raluca Ada Popa, Joseph E Gonzalez, and Ion Stoica. Helen: Maliciously secure coopetitive learning for linear models. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 724–738. IEEE, 2019.

[142] Ivan Damgård, Valerio Pastro, Nigel Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In *Annual Cryptology Conference*, pages 643–662. Springer, 2012.

[143] Andreea B Alexandru, Anastasios Tsiamis, and George J Pappas. Encrypted distributed lasso for sparse data predictive control. *arXiv preprint arXiv:2104.11632*, 2021.

[144] Wenting Zheng, Ryan Deng, Weikeng Chen, Raluca Ada Popa, Aurojit Panda, and Ion Stoica. Cerebro: A platform for multi-party cryptographic collaborative learning. In *30th {USENIX} Security Symposium ({USENIX} Security 21)*, 2021.

[145] Hao Chen, Wei Dai, Miran Kim, and Yongsoo Song. Efficient multi-key homomorphic encryption with packed ciphertexts with application to oblivious neural network inference. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 395–412, 2019.

[146] Jack LH Crawford, Craig Gentry, Shai Halevi, Daniel Platt, and Victor Shoup. Doing real work with fhe: the case of logistic regression. In *Proceedings of the 6th Workshop on Encrypted Computing & Applied Homomorphic Cryptography*, pages 1–12, 2018.

[147] Ehsan Hesamifard, Hassan Takabi, and Mehdi Ghasemi. Deep neural networks classification over encrypted data. In *Proceedings of the Ninth ACM Conference on Data and Application Security and Privacy*, pages 97–108, 2019.

[148] Dan Boneh, Kevin Lewi, Mariana Raykova, Amit Sahai, Mark Zhandry, and Joe Zimmerman. Semantically secure order-revealing encryption: Multi-input functional encryption without obfuscation. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 563–594. Springer, 2015.

[149] Brent Waters. A punctured programming approach to adaptively secure functional encryption. In *Annual Cryptology Conference*, pages 678–697. Springer, 2015.

[150] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. *SIAM Journal on Computing*, 45(3):882–929, 2016.

[151] Brent Carmer, Alex J Malozemoff, and Mariana Raykova. 5gen-c: multi-input functional encryption and program obfuscation for arithmetic circuits. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 747–764, 2017.

[152] Kevin Lewi, Alex J Malozemoff, Daniel Apon, Brent Carmer, Adam Foltzer, Daniel Wagner, David W Archer, Dan Boneh, Jonathan Katz, and Mariana Raykova. 5gen: A framework for prototyping applications using multilinear maps and matrix branching programs. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 981–992, 2016.

[153] Shweta Agrawal, David Mandell Freeman, and Vinod Vaikuntanathan. Functional encryption for inner product predicates from learning with errors. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 21–40. Springer, 2011.

[154] Allison Bishop, Abhishek Jain, and Lucas Kowalczyk. Function-hiding inner product encryption. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 470–491. Springer, 2015.

[155] Joël Alwen, Manuel Barbosa, Pooya Farshim, Rosario Gennaro, S Dov Gordon, Stefano Tessaro, and David A Wilson. On the relationship between functional encryption, obfuscation, and fully homomorphic encryption. In *IMA International Conference on Cryptography and Coding*, pages 65–84. Springer, 2013.

[156] Jérémy Chotard, Edouard Dufour Sans, Romain Gay, Duong Hieu Phan, and David Pointcheval. Decentralized multi-client functional encryption for inner product. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 703–732. Springer, 2018.

[157] Jérémy Chotard, Edouard Dufour-Sans, Romain Gay, Duong Hieu Phan, and David Pointcheval. Dynamic decentralized functional encryption. *IACR Cryptology ePrint Archive*, 2020.

[158] Runhua Xu and James Joshi. Revisiting secure computation using functional encryption: Opportunities and research directions. *arXiv preprint arXiv:2011.06191*, 2020.

[159] Théo Ryffel, Edouard Dufour Sans, Romain Gay, Francis Bach, and David Pointcheval. Partially encrypted machine learning using functional encryption. *arXiv preprint arXiv:1905.10214*, 2019.

[160] Chengliang Zhang, Suyi Li, Junzhe Xia, Wei Wang, Feng Yan, and Yang Liu. Batchcrypt: Efficient homomorphic encryption for cross-silo federated learning. In *2020 {USENIX} Annual Technical Conference ({USENIX}{ATC} 20)*, pages 493–506, 2020.

[161] Wilko Henecka, Stefan K ögl, Ahmad-Reza Sadeghi, Thomas Schneider, and Immo Wehrenberg. Tasty: tool for automating secure two-party computations. In *Proceedings of the 17th ACM conference on Computer and communications security*, pages 451–462, 2010.

[162] Brian Knott, Shobha Venkataraman, Awni Hannun, Shubho Sengupta, Mark Ibrahim, and Laurens van der Maaten. Crypten: Secure multi-party computation meets machine learning. In *Proceedings of the NeurIPS Workshop on Privacy-Preserving Machine Learning*, 2020.

[163] Sameer Wagh, Shruti Tople, Fabrice Benhamouda, Eyal Kushilevitz, Prateek Mittal, and Tal Rabin. Falcon: Honest-majority maliciously secure framework for private deep learning. *Proceedings on Privacy Enhancing Technologies*, 2021(1):188–208, 2020.

[164] Yi Li and Wei Xu. Privpy: General and scalable privacy-preserving data mining. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1299–1307, 2019.

[165] Xiaoyu Fan, Guosai Wang, Kun Chen, Xu He, and Wei Xu. Ppca: Privacy-preserving principal component analysis using secure multiparty computation (mpc). *arXiv preprint arXiv:2105.07612*, 2021.

[166] Sijun Tan, Brian Knott, Yuan Tian, and David J Wu. Cryptgpu: Fast privacy-preserving machine learning on the gpu. *arXiv preprint arXiv:2104.10949*, 2021.

[167] Javad Ghareh Chamani and Dimitrios Papadopoulos. Mitigating leakage in federated learning with trusted hardware. *arXiv preprint arXiv:2011.04948*, 2020.

[168] Andrew Law, Chester Leung, Rishabh Poddar, Raluca Ada Popa, Chenyu Shi, Octavian Sima, Chaofan Yu, Xingmeng Zhang, and Wenting Zheng. Secure collaborative training and inference for xgboost. In *Proceedings of the 2020 Workshop on Privacy-Preserving Machine Learning in Practice*, pages 21–26, 2020.

[169] Frank McKeen, Ilya Alexandrovich, Alex Berenzon, Carlos V Rozas, Hisham Shafi, Vedvyas Shanbhogue, and Uday R Savagaonkar. Innovative instructions and software model for isolated execution. In *Proceedings of the 2nd International Workshop on Hardware and Architectural Support for Security and Privacy*, HASP '13, New York, NY, USA, 2013. Association for Computing Machinery.

[170] David Kaplan, Jeremy Powell, and Tom Woller. Amd memory encryption. *White paper*, 2016.

[171] Xiao Shaun Wang, Kartik Nayak, Chang Liu, TH Hubert Chan, Elaine Shi, Emil Stefanov, and Yan Huang. Oblivious data structures. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 215–226, 2014.

[172] Emil Stefanov, Marten Van Dijk, Elaine Shi, Christopher Fletcher, Ling Ren, Xiangyao Yu, and Srinivas Devadas. Path oram: an extremely simple oblivious ram protocol. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 299–310, 2013.

[173] Pau-Chen Cheng, Kevin Eykholt, Zhongshu Gu, Hani Jamjoom, KR Jayaram, Enriquillo Valdez, and Ashish Verma. Separation of powers in federated learning. *arXiv preprint arXiv:2105.09400*, 2021.

[174] Chengliang Zhang, Junzhe Xia, Baichen Yang, Huancheng Puyang, Wei Wang, Ruichuan Chen, Istemi Ekin Akkus, Paarijaat Aditya, and Feng Yan. Citadel: Protecting data privacy and model confidentiality for collaborative learning with sgx. *arXiv preprint arXiv:2105.01281*, 2021.

[175] Fan Mo, Hamed Haddadi, Kleomenis Katevas, Eduard Marin, Diego Perino, and Nicolas Kourtellis. Ppfl: Privacy-preserving federated learning with trusted execution environments. *arXiv preprint arXiv:2104.14380*, 2021.

[176] Ping Li, Jin Li, Zhengan Huang, Chong-Zhi Gao, Wen-Bin Chen, and Kai Chen. Privacy-preserving outsourced classification in cloud computing. *Cluster Computing*, 21(1):277–286, 2018.

[177] Reza Shokri and Vitaly Shmatikov. Privacy-preserving deep learning. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, pages 1310–1321. ACM, 2015.

[178] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3):50–60, 2020.

[179] Yangsibo Huang, Yushan Su, Sachin Ravi, Zhao Song, Sanjeev Arora, and Kai Li. Privacy-preserving learning via deep net pruning. *arXiv preprint arXiv:2003.01876*, 2020.

[180] Boxin Wang, Fan Wu, Yunhui Long, Luka Rimanic, Ce Zhang, and Bo Li. Datalens: Scalable privacy preserving training via gradient compression and aggregation. *arXiv preprint arXiv:2103.11109*, 2021.

[181] Yue Wang, Cheng Si, and Xintao Wu. Regression model fitting under differential privacy and model inversion attack. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[182] Ligeng Zhu and Song Han. Deep leakage from gradients. In *Federated Learning*, pages 17–31. Springer, 2020.

[183] Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. Inverting gradients–how easy is it to break privacy in federated learning? *arXiv preprint arXiv:2003.14053*, 2020.

[184] Zaoxing Liu, Tian Li, Virginia Smith, and Vyas Sekar. Enhancing the privacy of federated learning with sketching. *arXiv preprint arXiv:1911.01812*, 2019.

[185] Jing Ma, Si-Ahmed Naas, Stephan Sigg, and Xixiang Lyu. Privacy-preserving federated learning based on multi-key homomorphic encryption. *arXiv preprint arXiv:2104.06824*, 2021.

[186] Sameer Wagh, Xi He, Ashwin Machanavajjhala, and Prateek Mittal. Dp-cryptography: marrying differential privacy and cryptography in emerging applications. *Communications of the ACM*, 64(2):84–93, 2021.

[187] Hanieh Hashemi, Yongqin Wang, Chuan Guo, and Murali Annavaram. Byzantine-robust and privacy-preserving framework for fedml. *arXiv preprint arXiv:2105.02295*, 2021.

[188] Yuhui Zhang, Zhiwei Wang, Jiangfeng Cao, Rui Hou, and Dan Meng. Shufflefl: gradient-preserving federated learning using trusted execution environment. In *Proceedings of the 18th ACM International Conference on Computing Frontiers*, pages 161–168, 2021.

[189] Yehuda Lindell and Benny Pinkas. Privacy preserving data mining. In *Annual International Cryptology Conference*, pages 36–54. Springer, 2000.

[190] Eugene Bagdasaryan, Omid Poursaeed, and Vitaly Shmatikov. Differential privacy has disparate impact on model accuracy. In *Advances in Neural Information Processing Systems*, pages 15453–15462, 2019.

[191] Ali Shafahi, W Ronny Huang, Mahyar Najibi, Octavian Suciu, Christoph Studer, Tudor Dumitras, and Tom Goldstein. Poison frogs! targeted clean-label poisoning attacks on neural networks. In *Advances in Neural Information Processing Systems*, pages 6103–6113, 2018.

[192] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733*, 2017.

[193] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How to backdoor federated learning. *arXiv preprint arXiv:1807.00459*, 2018.

[194] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

[195] Lei Yu, Ling Liu, Calton Pu, Mehmet Emre Gursoy, and Stacey Truex. Differentially private model publishing for deep learning. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 332–349. IEEE, 2019.

[196] Rachel Cummings, Varun Gupta, Dhamma Kimpara, and Jamie Morgenstern. On the compatibility of privacy and fairness. In *Adjunct Publication of the 27th Conference on User Modeling, Adaptation and Personalization*, pages 309–315, 2019.

[197] Mohammad Naseri, Jamie Hayes, and Emiliano De Cristofaro. Toward robustness and privacy in federated learning: Experimenting with local and central differential privacy. *arXiv preprint arXiv:2009.03561*, 2020.

[198] Matthew Jagielski, Jonathan Ullman, and Alina Oprea. Auditing differentially private machine learning: How private is private sgd? *arXiv preprint arXiv:2006.07709*, 2020.

[199] Yuzhe Ma, Xiaojin Zhu, and Justin Hsu. Data poisoning against differentially-private learners: Attacks and defenses. *arXiv preprint arXiv:1903.09860*, 2019.

[200] Sebastian Caldas, Sai Meher Karthik Duddu, Peter Wu, Tian Li, Jakub Konečnỳ, H Brendan McMahan, Virginia Smith, and Ameet Talwalkar. Leaf: A benchmark for federated settings. *arXiv preprint arXiv:1812.01097*, 2018.