Private and Secure Distributed Deep Learning: A Survey

CORINNE ALLAART, Vrije Universiteit Amsterdam, Amsterdam, Netherlands and St. Antonius Ziekenhuis, Nieuwegein, Netherlands
SABA AMIRI, Universiteit van Amsterdam, Amsterdam, Netherlands
HENRI BAL, Vrije Universiteit Amsterdam, Amsterdam, Netherlands
ADAM BELLOUM, FNWI, Universiteit van Amsterdam, Amsterdam, Netherlands
LEON GOMMANS, Koninklijke Luchtvaart Maatschappij, Amsterdam, Netherlands
AART VAN HALTEREN, Vrije Universiteit Amsterdam, Amsterdam, Netherlands and Philips Research, Eindhoven, Netherlands
SANDER KLOUS, Universiteit van Amsterdam, Amsterdam, Netherlands

Traditionally, deep learning practitioners would bring data into a central repository for model training and inference. Recent developments in distributed learning, such as federated learning and deep learning as a service (DLaaS), do not require centralized data and instead push computing to where the distributed datasets reside. These decentralized training schemes, however, introduce additional security and privacy challenges. This survey first structures the field of distributed learning into two main paradigms and then provides an overview of the recently published protective measures for each. This work highlights both secure training methods as well as private inference measures. Our analyses show that recent publications, while being highly dependent on the problem definition, report progress in terms of security, privacy, and efficiency. Nevertheless, we also identify several current issues within the private and secure distributed deep learning (PSDDL) field that require more research. We discuss these issues and provide a general overview of how they might be resolved.

CCS Concepts: • **Computing methodologies → Distributed artificial intelligence**; **Neural networks**; • **Security and privacy**;

Additional Key Words and Phrases: Deep learning, privacy, security, distributed learning

ACM Reference Format:
Corinne Allaart, Saba Amiri, Henri Bal, Adam Belloum, Leon Gommans, Aart van Halteren, and Sander Klous. 2024. Private and Secure Distributed Deep Learning: A Survey. *ACM Comput. Surv.* 57, 4, Article 92 (December 2024), 43 pages. https://doi.org/10.1145/3703452

Corinne Allaart and Saba Amiri contributed equally to this work.

Authors' Contact Information: Corinne Allaart, Vrije Universiteit Amsterdam, Amsterdam, Netherlands and St. Antonius Ziekenhuis, Nieuwegein, Netherlands; e-mail: c.g.allaart@vu.nl; Saba Amiri, Universiteit van Amsterdam, Amsterdam, Noord-Holland, Netherlands; e-mail: s.amiri@uva.nl; Henri Bal, Vrije Universiteit Amsterdam, Amsterdam, Noord-Holland, Netherlands; e-mail: h.e.bal@vu.nl; Adam Belloum, FNWI, Universiteit van Amsterdam, Amsterdam, Noord-Holland, Netherlands; e-mail: a.s.z.belloum@uva.nl; Leon Gommans, Koninklijke Luchtvaart Maatschappij, Amsterdam, Netherlands; e-mail: leon.gommans@klm.com; Aart van Halteren, Vrije Universiteit Amsterdam, Amsterdam, Noord-Holland, Netherlands and Philips Research, Eindhoven, North Brabant, Netherlands; e-mail: a.t.van.halteren@vu.nl; Sander Klous, Universiteit van Amsterdam, Amsterdam, Noord-Holland, Netherlands; e-mail: s.klous@uva.nl.

ACM 0360-0300/2024/12-ART92
https://doi.org/10.1145/3703452

1 Introduction

The ubiquitous popularity of **Deep Learning (DL)** in diverse applications in different domains, and the fact that these methods generally benefit from large amounts of data as training input, has led to an increased need to access big data reservoirs. This need continues beyond the availability of central databases, as the quantity of data necessary to train deep models in certain use cases can be higher than many centralized datasets alone can provide. Moreover, the age of big data leads to data generation being a normal function of entities in different scales, from individuals to large organizations. Creating extensive centralized databases can be costly, problematic in terms of privacy, or simply infeasible due to processing and communication restrictions. This has led to an interest in **Distributed Deep Learning (DDL)**, where multiple parties cooperate to utilize distributed data for a combined DL effort.

Distributed learning covers a wide set of paradigms, such as **Machine Learning as a Service (MLaaS)**, which makes models available for inference on new distributed samples, and **federated learning (FL)**, which allows for the collaborative training of a single neural network on distributed local datasets. In distributed deep learning, the problem of decentralized data access is resolved by moving computing to where the data resides. However, additional problems arise in the DDL context. One of the most prominent challenges is retaining the privacy and security of data and models in distributed settings. With the cooperation of multiple different parties, there are often incentives to keep local data and/or models private, be it for legal regulations such as GDPR [44], an unwillingness to share to keep a competitive advantage towards other parties, or ethical concerns. This has amplified interest in the application of privacy- and security-preserving mechanisms during model development, testing, implementation, and inference.

We define the privacy-preserving paradigm of distributed deep learning in two different contexts: **Privacy** and **Security**. Security is defined as the safeguarding of data and models. Achieving secure DDL means a malicious party having access to the DDL process would not be able to gain unauthorized access to training data, model parameters, communications, and so on. Privacy, however, is defined as stopping the models, either partially trained local models or fully trained centralized ones, from leaking information about the training data and the model itself. In this article, we refer to security as the classification of the methods that protect against unauthorized access to data, mainly through cryptographic means. Common types of cryptographic methods are **Secure Multiparty Computation (MPC)**, which provides secure messaging protocols between different parties, and **Homomorphic Encryption (HE)**, a type of encryption that allows accurate calculations to be performed on encrypted data. The privacy front in this work aims for methods that provide formal guarantees of privacy with quantifiable privacy levels and therefore focuses on methods that provide **Differential Privacy (DP)** guarantees.

Secure and private methods of DDL have been well-researched. Several surveys have previously attempted to structure and summarize studies performed in this research area [41, 129, 150, 157]. Most of these reviews only focus on one class of protective measures, such as differential privacy [5] or multiparty computation [157], or on a subset of the available types of distributed learning, such as MLaaS [129] or Federated Learning [41, 150]. In this work, we aim to provide a larger and more comprehensive overview of the field of **private and secure distributed deep learning (PSDDL)**, covering both the security and privacy aspects of these methods. In this way, this survey serves as a starting point for comparisons between the different privacy and security methods and evaluations of applicable protective measures in different use cases. Depending on the use case requirements, including the distribution of data and models, the level of security of privacy required, and the possibility of a central server, the requirements of the security- and privacy-preserving methods change. Therefore, this survey aims to provide a high-level overview of the field of private and secure distributed deep learning and to discuss the possibilities of privacy and security mechanisms per type of distributed deep learning workflow.

The structure of the article is thus as follows: Section 2 shows our survey protocol, where we define the scope of the survey. In Section 3, we describe our taxonomy along which we organized the selected papers. We divided the type of distribution workflow into two main categories. Sections 4 and 5 discuss the papers applicable to the two main DDL categories, respectively, **collaborative learning (CBL)** and distributed learning. Both the categories will focus on centralized workflows, but the discussion (Section 6) will also touch upon decentralized frameworks.

2 Survey Protocol

2.1 Scope of Survey

The scope of this survey is *private and secure distributed deep learning*. We explain the choice of this scope by explaining its following parts: Private and Secure, Distributed, and Deep Learning.

Private and Secure: In this survey, we make a distinction between security and privacy of distributed DL models. The security aspect of these methods is concerned with stopping adversarial behavior, both by members of the distributed pipeline as well as outside entities, from having unauthorized access to the training or model data and being able to potentially tamper with them. For privacy, we look at how to reduce and/or mitigate the risk of information leakage on the inference surface of the model and protect it against adversarial attacks. Privacy is mainly achieved by differential privacy methods, while cryptographic methods are the main mechanisms for security. While their methods and definitions are different, their goals are complementary, and both need to be considered for properly safe architecture.

Distributed: We aim for this survey to serve as a starting point for recommendations of security and privacy depending on the use case requirements, including the distribution of data and models, the level of security of privacy required, and the possibility of a central server, the requirements of the security- and privacy-preserving methods change. Therefore, it is essential to discuss different types of data distribution. Our comprehensive view explains the strengths and vulnerabilities of different types of distributed learning and how different protective measures help to reach the necessary privacy and security depending on the use case.

Deep Learning: We chose to limit the survey to Deep Learning instead of all **Machine Learning (ML)**. DL offers state-of-the-art performance for most tasks, especially on tasks with high-volume data, which is often the subject of distributed learning. Moreover, the security and privacy challenges can be different from other ML models, as the complexity of the models increases the chances of private information being extracted from DL models. For brevity and clarity, we therefore only focus on DL.

The combination of these three aspects of this survey offers us the opportunity to provide a high-level and more comprehensive overview of the field of privacy and secure distributed deep learning by being able to compare not only within one sub-specialty, but among different types of security and privacy and distributed learning methodologies. To underline this, we compare the scope of our survey to other recent surveys in Table 1. We highlight that we did not find any surveys that focus on both collaborative learning and federated learning, as well as other distributed learning methods, such as split learning [134]. However, these methods are often used for similar purposes and therefore warrant comparison for a full overview of the field of PSDDL. Information on our selection criteria, including our search query, can be found in Appendix A.1.

3 PSDDL Taxonomy

To effectively compare the selected papers, in this work, we present a taxonomy based on the use case of the methods described in the papers. There are different configurations of Distributed Deep

Table 1. Comparison of Surveys within the Field of PSDDL

Reference	ML or DL	Collaborative Learning		Federated Learning		Other DDL methods	Attacks
		Cryptographic methods	Differential Privacy	Cryptographic methods	Differential Privacy		
[129]	DL	✓	✓				✓
[157]	DL	✓					
[150]	ML			✓	✓		
[41]	DL			✓	✓		✓
Our	DL	✓	✓	✓	✓	✓	

Learning in which security and privacy preserving mechanisms can be applied, which leads us to base our taxonomy on the workflow of secure and private DDL. As such, this article can serve as a guideline, showing which types of preservation mechanisms can be applied in different domains and for different use cases, depending on the intended DDL architecture.

3.1 The Distributed Deep Learning Framework

While many different types of DDL workflows have been developed, the principal areas of research in this context can be classified into two main categories: *Collaborative Learning* and *Federated Learning*.

Collaborative Learning (CBL) In this setup, the central server trains one central model, while the data is distributed over different clients. The local data and its derivatives are securely communicated to the server. A distinction here can be made between collaborative learning, where the distributed data is used for model training, and ***Secure Inference (SI)***, where inference by a previously trained model is performed on a local sample. The latter usually is the basis of **Deep Learning as a Service (DLaaS)**.

Distributed Learning Here, clients do not share their data but cooperatively train a model by sharing their private models' details, such as gradients and weights. The most common architecture is **federated learning (FL)**. In FL's basic form, clients train a local copy of the central model on their private datasets and share raw model parameters or their derivatives with an orchestrator. The orchestrator uses an aggregation mechanism to combine the received parameters and communicates that back to the participants. This process is iterated until convergence. We will also discuss other distributed learning methods, such as split learning [134] and ensemble methods. It is important to mention we use a slightly narrower definition of distributed learning to refer to methods that involve model sharing.

These categories are illustrated in Figure 1, with FL as an example of distributed learning. Within each main category, there are different points in the architecture where privacy or security mechanisms can be applied. Section 4 will discuss the developed methods for CBL, and Section 5 will cover DDL. Some workflows cannot be classified into these two categories, most notably decentralized distributed learning. We choose not to include these in the body of our survey, as the protective measures on these types of workflows are still in the emerging stage. In Section 6, we will highlight how they compare to the protective measures in our two main categories.

Data Partitioning in Distributed Learning There are different ways a dataset can be divided over different clients. The most commonly occurring distribution is horizontal partitioning, where every client has a partial dataset that contains a partition of the samples. In vertical partitioning, they contain a part of the features. Most research focuses on horizontally partitioned data. Therefore, we will specifically mention when a framework offers privacy protection for vertically partitioned distributed learning.

3.2 Privacy Vulnerabilities of Deep Learning Models against Privacy Attacks

Distributed and federated learning aim to decentralize model training, mitigating risks associated with centralized data storage. Both paradigms leverage multiple devices or servers holding local

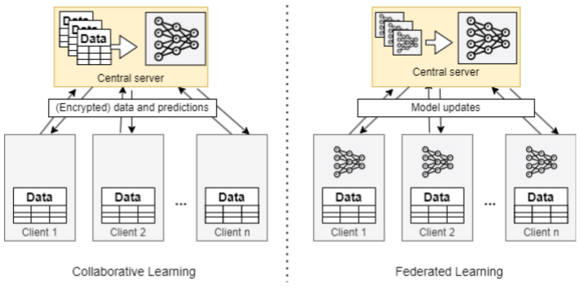

Fig. 1. Two major architectures for distributed learning. In FL, participants are actively involved in training the model, while in collaborative learning, participants contribute by sharing their data or results for DL training and inference on their data in a private and secure manner by an orchestrator

data samples, thereby enhancing privacy by minimizing raw data sharing. Distributed learning performs parallel computations on data distributed across various locations. Federated learning, a specialized form of distributed learning, employs orchestrators to facilitate training while data resides on user devices. Despite these decentralized approaches, both distributed and federated learning remain susceptible to privacy vulnerabilities during training and inference [87].

While essential for model convergence, the exchange of model updates in distributed and federated learning creates privacy vulnerabilities. Analyzing update patterns over multiple iterations can reveal sensitive information about local datasets [45]. Even during inference, model responses can leak information, potentially allowing adversaries to reconstruct a surrogate model and infer characteristics of the training data [114].

3.2.1 Privacy Attacks for Exploiting Machine Learning Models. Privacy attacks against DDL are methods by which malicious entities attempt to exploit vulnerabilities in these systems to gain unauthorized access to sensitive data. Privacy attacks generally exploit the data processing capabilities of distributed ML models and architectures to infer protected or sensitive information. These attacks can occur at various stages of a DDL model's lifecycle, including during collaborative training, inference, or data transmission phases. Below, we outline some general mechanisms often employed in privacy attacks on ML systems [80]. Based on their goals and their exploitation method, these adversarial attacks are categorized as *Membership Inference*, *Reconstruction*, *Inversion*, and *Property Inference* attacks.

Membership Inference Attacks. Membership inference attacks [63] are sophisticated privacy breaches aimed at deducing the presence or absence of specific data instances in the training set of an ML model. These attacks exploit differential behaviors exhibited by the model processing input data that it has encountered during training versus unseen data. The core hypothesis driving these attacks is that ML models tend to predict with higher confidence and lower entropy when presented with familiar data. Attackers leverage this by analyzing the confidence levels, decision boundaries, and output probabilities of the model for various inputs and can infer with some probability if a given data point was part of the training dataset. This type of attack poses significant risks, particularly in scenarios where the inclusion of an individual's data in a model's training set could reveal sensitive personal information.,

Reconstruction Attacks. Reconstruction attacks [95] focus on the capability to approximate or fully reconstruct the input data used to train a machine-learning model based solely on observing the model's outputs. These attacks meticulously probe the ML model with a series of strategically crafted inputs and analyze the corresponding outputs to deduce the characteristics or even exact entries of the training data. Attackers may employ techniques such as gradient-based optimization to modify their input iteratively, moving closer to the original training data with each step by minimizing the difference between their input's predicted outputs and the outputs observed when the actual training data is used. This attack vector is particularly alarming for models trained on confidential or proprietary data, as it can lead to significant privacy breaches or intellectual property theft.

Inversion Attacks. Inversion attacks [60] are a nuanced subset of reconstruction attacks where the objective is to invert the model's predictive function to reveal details about the training data or reconstruct the input leading to specific outputs. These attacks utilize the predictable nature of the model and the information encoded within its parameters to reverse-engineer the inputs. By understanding how model outputs change in response to variations in input and by exploiting the model's gradients, attackers can craft inputs that, when processed by the model, reveal substantial information about the data on which the model was trained. The challenge and effectiveness of inversion attacks depend significantly on the model's complexity and the amount of information implicitly stored in its parameters.

Property Inference Attacks. Property inference attacks [161] are designed to uncover global statistical properties of the dataset upon which a machine-learning model was trained, rather than focusing on individual data entries. These attacks exploit the model's learned characteristics to infer broader patterns, trends, or biases in the training data. For instance, an attacker might analyze the model's outputs across a range of inputs to determine if the training data over-represents a particular demographic or if certain predictive features are correlated with sensitive attributes. Property inference attacks typically involve complex statistical methods and hypothesis testing against the model's predictions to derive insights that might reveal sensitive or discriminatory practices in data handling and model training.

3.3 Protective Mechanisms against Adversarial Attacks

In this work, we make a distinction between security mechanisms and privacy-preserving methods. Within these two categories, many different methods are applied by the reviewed papers. To make this work approachable to a larger audience, we will give a short overview and explanation of these mechanisms in this section. Readers interested in detailed explanations and formal definitions of these methods are encouraged to consult the appendices.

3.3.1 Security Mechanisms. The type of security mechanisms reviewed in this work is based on the type of encryption protocols. In PPDDL, two main types of advanced cryptography are used: **Secure Multiparty Computation (SMPC)** and **Homomorphic Encryption (HE)**. Secure Multiparty Computation (SPMC) Multiple parties collaborate to perform a calculation while keeping their input in the calculation secret. We make a distinction between **multiparty computation (MPC)**, with three or more clients, and **two-party computation (2PC)**, with two clients. While various protocols have been developed to achieve SPMC, the most common method used is *garbled circuits*—which is based on oblivious transfer—for 2PC and *secret sharing* for MPC. These methods are further explained and discussed in the Appendix. Homomorphic encryption allows for calculations to be performed on encrypted data. When the results of these calculations are decrypted, it is as if the calculations were performed on the unencrypted data. In ***full homomorphic encryption (FHE)***, all types of calculations are possible. Using HE introduces noise, and FHE is only possible by often re-encrypting the data to limit the noise. Doing these calculations can become very computationally expensive. There are several ways this cost can be reduced. One possibility is to limit the type of allowed computations, such as with **additional homomorphic encryption (AHE)**, which only allows for addition-based computations. Another possibility is to retain the option for all calculations but maintain a limited depth of the circuit, as with *Leveled FHE*. While

LFHE allows for both multiplication and addition (essentially the basis of all calculations) there is a limited depth, which might not allow for complicated functions such as the activation function in a neural network. It is important to note that HE offers honest-but-curious privacy protection.

3.3.2 Security Levels. When promoting the privacy preservation of a protocol or workflow, the level of adversary it has been proven to withstand needs to be addressed. This includes both their access level and their behavior. These classifications, security levels, determine in what type of security attacks a network would remain secure. The two main classifications that are covered are semi-honest or ***honest-but-curious (HBC)*** and ***Malicious Adversary (MA)***. In both situations, adversaries are trying to collect secure information. In the HBC scenario, the adversaries will try to get as much information as possible but will not break the rules of the protocol. In the MA scenario, the adversary will also break protocol wherever possible. Another classification of security levels is the collaboration or collusion of adversaries, as a setting where multiple adversaries are cooperating to gain private information needs a more secure protocol. In the MA scenario, there is also robustness and fairness to consider: When a malicious party is discovered, the standard MA protocol will simply be interrupted. In a robust or fair scenario, either all or no parties receive the outcome, respectively.

3.3.3 Privacy Mechanisms. We define the privacy of ML models as preventing them from leaking information about their training data and their internal information and state. In a centralized setting, this only concerns the inference API of the model. In a distributed and/or federated setup, we also need to consider the privacy of the output of the participants and the orchestrator as well as the privacy of the final model(s) once their inference API is released. Different techniques have been proposed to preserve the privacy of randomized mechanisms such as ML models, including different levels of anonymization. However, it has been shown that these methods are susceptible to adversarial attacks such as linkage attacks to certain degrees [107].

Differential Privacy (DP) is a standard for preserving the privacy of randomized mechanisms that provides a rigorous mathematical proof of privacy. In its most extreme case, it provides plausible deniability to any individual member of the training set if our ML model is trained on two datasets that differ in only that member and the adversary knows all about the difference in the datasets and can observe the distribution of the output of our model. DP will limit the difference in the log-likelihood of the same model trained on these two *adjacent* datasets by a privacy budget ϵ. There are different definitions for DP, but ϵ-DP, originally introduced by Dwork et al. [39], only considers the privacy budget. The lower the privacy budget is, the higher the level of privacy of the model would be. This comes at a cost of performance and potential fairness for the model, and a tradeoff needs to be considered between the privacy and utility of the model. Differential privacy is typically achieved by introducing some type of perturbation at training and/or inference time, usually through adding carefully calibrated noise. The aim of the noise-adding mechanism is to add enough perturbation to the system so the model learns as little as possible about individual members of the training set while learning as much as possible about the population. More details on the definition of DP and its noise-adding mechanisms can be found in the Appendix.

3.3.4 Privacy Levels. In a typical centralized distributed learning scenario such as federated learning depending on the threat model, different DP levels can be achieved. Generally, we can add DP on *local* and *global* levels. Local DP adds noise-adding mechanisms—and thus ensures DP guarantees—on participant levels while global DP uses DP mechanisms to ensure privacy on the parameterset level on the orchestrator. Global DP is desirable when the orchestrator is trusted but there are no trust assumptions about the participants. Local DP is used when the orchestrator is not trusted but no expectation of adversarial behavior is included in the threat model. Combining

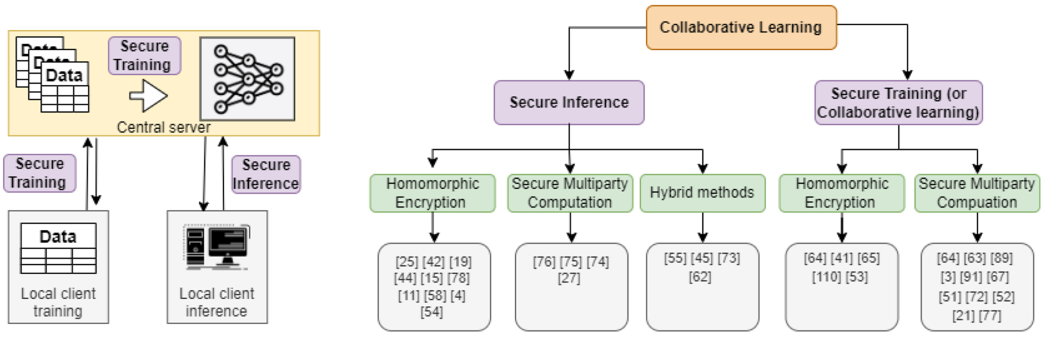

Fig. 2. Left: CBL workflow. It shows at which point secure inference, secure training, and private server training are applied. Right: Distribution of papers among the different types of protective mechanisms for the different points in the workflow.

local and global DP ensures privacy in a most pessimistic threat model with both the orchestrator and the participants being expected to behave in an adversarial manner. Although, reiterating our earlier point, adding additional levels of privacy usually comes at the price of loss of utility—although this assumption is not unequivocally true, as DP noise-adding mechanisms have been shown to act as regularization mechanisms that would benefit certain aspects of the model utility; details of this process are out of the scope of this review.

4 Collaborative Learning

In **collaborative learning (CBL)**, the distributed data of the clients is sent to a central server, where the data is used for machine learning. Thus, the local clients do not build a local model themselves. The workflow of CBL is shown in Figure 2 (left). As shown, there are three points where protective measures can be taken:

(1) during the exchange of the training set data;
(2) during the training of the model on the server;
(3) during the exchange of the sample and subsequent label for inference.

If there is a previously trained model, then the clients often may only look to perform the last step of the workflow: inference. The trained model is stored in a server and clients have unlabeled samples that they want to classify. An example would be DLaaS, where the service provider aims to keep their model private, and the client aims to keep their data private, yet both parties want the model to perform inference on the data. These situations are referred to as **secure inference (SI)**, as will be discussed in Section 4.2 In Section 4.3, the papers are included that focus on secure training, the training phase of CBL. In those situations, next to the security-preserving mechanisms applied to the exchanging of partial datasets or a single samples for inference, the server might also apply differential privacy when training the model on the server, which is referred to as step 2 in the workflow. We exclude privacy-preserving methods through differential privacy in this section. As the parties do not perform their own training or analysis, there is no special setting for DP. We therefore refer to other surveys on local DP, such as Reference [54] Figure 2 (right) gives an overview and categorization of the discussed papers in this chapter.

4.1 Secure Inference

This section will discuss the selected frameworks that focus on facilitating secure inference. In these cases, there is an already trained network that the server provides for inference of the client's unseen samples and the data of the client should remain private. The server would also like to

Table 2. Overview of Secure Inference Methods

Framework	Name	Encryption Mechanism	Threat Level		Type of NN	Activation function	Evaluation Datasets	Results on MNIST [31]		Verif.	Novelty
			Level	Col				Accuracy (%)	efficiency		
[35]	CryptoNets	FHE (YASHE)	HBC	ND	CNN	polynomial	MNIST	98.95	297.65 s	–	First paper to use HE for SI
[15]	DiNN	TFHE	HBC	ND	MLP	polynomial	MNIST	96	1.7 s	–	Discretized neural networks
[62]	CryptoDL	FHE (BGV)	HBC	ND	CNN	polynomial	MNIST CIFAR10[74]	99.5	2.1x to [35]	–	Low degree polynomial approximations of activation functions
[81]	MiniONN	GC SS AHE	HBC	ND	CNN LSTM	exact	MNIST CIFAR10 a.o.	99	1.3s	–	Introduces oblivious neural networks
[113]	Chameleon	SS GMW GC	HBC	NC	MLP CNN	exact	MNIST	99	2.7s	–	Combined MPC methods for efficiency, using a trusted third party
[26]	Faster CryptoNets	FHE (FV-RNS)	HBC	ND	CNN	polynomial	MNIST CIFAR10	99	46s	–	Speeds up CryptoNets while adding DP
[116]	DeepSecure	GC OT	HBC	ND	CNN DNN	exact	MNIST a.o.	99	438 to [35]	–	Efficient SI based on GC
[66]	Gazelle	GC SS LHE (BFV)	HBC	ND	CNN	exact	MNIST CIFAR10	ND	0.03s	–	Uses garbled circuits to linearize inference for LHE
[119]	Tapas	TFHE	HBC	ND	CNN	binary	MNIST diabetes[126] a.o.	99	147s	–	Uses TFHE to implement binary NNs
[11]	NGraphHe2	FHE (CKKS)	HBC	ND	CNN	polynomial	MNIST	99	2s	–	Client-aided activation function computation
[112]	XONN	GC OT	HBC	NC	CNN	exact	MNIST CIFAR10 a.o.	99	1.28s	–	Replaces multiplication with XNOR for low-cost OT
[92]	Delphi	GC LHE (BFV)	HBC	ND	CNN	polynomial	CIFAR10 CIFAR100[74]	ND	ND	–	Combines GC and LHE with planner for efficiency
[110]	Cheetah	LHE (BFV)	HBC	ND	NN CNN	polynomial	ImageNet[30] MNIST	ND	79x to [66]	–	Tuning and operator schedule optimizations for speedup, with GPUs
[3]	HCNN	FHE (BFV)	HBC	ND	CNN	polynomial	MNIST CIFAR10	99	5.2s	–	GPU-accelerated FHE
[85]	–	FHE (CKKS, SEAL)	HBC	ND	DNN CNN	polynomial	ImageNet	ND	ND	–	Uses hybrid memory systems to run large neural networks
[37]	PVDLI	SS	HBC	NC	CNN	polynomial	MNIST CIFAR10	98.1	2.5s	Y	Verifiability of the inference using SS
[76]	–	FHE (RNS CKSS)	HBC	ND	CNN	polynomial	CIFAR10	–	–	–	Polynomial activation functions for deep models

The papers are sorted by year of publication and by alphabetical order of the author. Collusion: whether participants can collude with the server. Verification: whether the paper addresses verifiability of the data.
Abbreviations: AHE= additional homomorphic encryption, BFV=Brakerski-Fan-Vercauteren scheme, CKKS= Cheon-Kim-Kim-Song scheme, CNN= convolutional neural network, DNN= deep neural network FHE= Full Homomorphic Encryption, GC= Garbled Circuits, GMW= Goldreich-Micali-Wigderson scheme, HBC = honest but curious, LSTM= Long short-term memory, LHE= Linearly Homomorphic Encryption, MA= malicious adversary, MLP= multi-layer perceptron, NC= No collusion, ND= not determined, OT= Oblivious transfer, SS= secret sharing.

keep its neural network secure, because of issues such as data leakage, or, in a situation such as DLaaS where the model in the product they are providing, to keep a competitive advantage. All of the discussed frameworks that facilitate secure inference used encryption-based security methods, although a few combined them with differential privacy. The different frameworks used HE, MPC, or a combination of the two. In the following sections, the papers will be discussed organized by encryption method. Section 4.1.1 focuses on HE, Section 4.1.2 on MPC, and Section 4.1.3 on hybrid methods. The SI frameworks are summarized in Table 2.

4.1.1 Secure Inference with Homomorphic Encryption. Several papers had an idea of using encrypted data for machine learning purposes [7, 53], but the first to develop a system that allowed for deep learning inference of homomorphically encrypted data was Reference [35] with CryptoNets. They applied leveled HE using the YASHE scheme to neural networks. This YASHE scheme has since been shown to be vulnerable [4, 71], and later papers will use different schemes. Nevertheless, they laid the foundation for secure inference.

One of the main issues with inference over homomorphically encrypted data is the activation function of neural networks. It decides whether a certain neuron should be activated by calculating the weighted sum of the inputs to the neuron and adding bias to it. The most common activation functions are ReLU, Sigmoid, and Tanh. The whole purpose of the activation functions is to introduce non-linearity into the outputs of the neurons, for them to be able to perform more complex tasks. Therefore, the activation functions are non-linear and complex functions. Unfortunately, those are the types of functions that the commonly applied leveled FHE schemes have trouble with, as they support addition and multiplication, but only up until a set calculation depth. The standard activation function would be too "deep" in that regard. As the approximation of the activation function is a main bottleneck in the implementation of HE-based secure inference, it is a main topic in the field. One way to circumvent this complexity problem is to provide a different activation function that is an approximation of the original one, but less complex, or more suitable for homomorphic encryption. As this is an approximation, it can lead to a loss of predictive performance. This is what CryptoNets introduced, and several other papers in this section thereafter aimed to improve. Another option is to use a cryptographic protocol to replace to activation function, these papers will be discussed in the hybrid section.

With CryptoNets, they approximated the sigmoid activation by replacing the max pooling step i with a scaled mean-pool square function. While HE-suitable, it resulted in the computing costs increasing with the number of NN layers and for bigger networks eventually becoming prohibitive. CryptoDL by Reference [62] builds on CryptoNets by adding approximations of several new activation functions. It implements low-degree polynomial approximations of several activation functions (i.e., ReLU, Sigmoid, and Tanh). The use of their approximates of activation functions leads to a closer approximation of a "normal" NN, which is visible in their performance.

Several other papers improved on other aspects of the CryptoNets framework. FasterCryptoNets [26] aims to provide a speedup in efficiency on CryptoNets. It not only increases efficiency, thus speeding up the inference, but it also provides an option for differential privacy for a small cost in predictive performance. Reference [65] focuses on the addition of a batch normalization to the scheme. Batch normalization is a technique that converts interlayer outputs into a standard format, which allows the layers to learn independently of the others. The purpose of the paper is to create a normalization that is reparametrized to support the conditions of the leveled FHE scheme.

References [15, 119] take a different approach to improve the efficiency of the secure inference framework. As mentioned in Section 3, there are different HE schemes with different properties. One main difference is the type of data input they can handle. While the previous frameworks used schemes that supported long values, References [15, 119] used TFHE, a scheme that only allows for binary operations. This makes the values used in the networks less precise, as they need to be binarized, but this HE scheme is more efficient than the other schemes. Reference [15] developed FHE-DiNN, a discretized network whose complexity is strictly linear to the depth of the network. Reference [119] also implemented TFHE in TAPAS, thereby allowing only binary weights and activations. The framework is implemented as a **Binary Neural Network (BNN)**, by allowing for XOR, AND, and XNOR operations through a function called reduce tree. While efficient and secure, the binary NN does lead to an, albeit small, decrease in predictive performance, compared to other HE frameworks. More recent papers mainly focus on improving the speedup of the networks to allow for larger networks to be run as secure inference.

The frameworks in References [35, 62] have shown that the predictive performance of SI is comparable to unencrypted inference. The following papers show speedup through a compiler [11], GPUs [3], and different approximation functions [76]. Reference [11] introduces their framework based on their previously developed NGraphHE compiler [12]. They leverage the CKKS FHE scheme, which offers support for real numbers. They introduce a client-aided model, where the two parties can cooperatively compute activation functions. Thereby, they implemented the largest network at the time, pre-trained MobileNetV2, and ran inference on the ImageNet dataset. This is extended in Reference [85], where hybrid memory is used to implement even larger models, such as ResNet-50. Reference [3] improves efficiency by presenting the first GPU-accelerated FHE, creating a speedup of the classification process. This allows for the classification of large batches of images. Reference [76] proposes a new method of approximation of the activation function. It replaces the ReLU and max-pooling functions of other activation functions, with a composition of min-max approximate polynomials of small degrees. The small degrees allow for more efficiency while it allows for common and well-developed deep learning models, as shown for ResNet and VGGNet, to be implemented without further modification and reusing of the pre-trained parameters.

4.1.2 Secure Inference with MPC. While the first frameworks of secure deep learning inference were developed with HE, several papers focused on implementing a system with MPC methods. MPC approaches are generally faster, but they do usually rely on all parties being involved in the whole computation. This can be problematic, but not necessarily in a two-party inference setup, as is common in secure inference. Often a combination of different MPCs was used for different steps in the neural network, for the different protocols have different characteristics suitable for different types of calculations. DeepSecure, by Reference [116], chose to diverge from homomorphic encryption by using **Garbled Circuits (GC)** to create a provably secure inference framework. It achieves this by transforming the input data into a "lower-dimensional subspace ensemble." Moreover, it converts and approximates the neural network into XOR, AND, XNOR gates that allow for the garbled circuit protocol. It is provably secure and offers a speedup compared to CryptoNets.

Reference [113] uses a hybrid system of MPC to create a more efficient system for secure inference. It combines both 2PC and MPC techniques. In particular, it performs linear operations using additively secret shared values and nonlinear operations using Garbled Circuits or the Goldreich-Micali-Wigderson protocol [52]. Moreover, Chameleon's framework moves away from the common assumption of secret sharing models that three or more parties need to communicate in the online phase, and it reduces communication overhead by precomputing heavy cryptographic operations in the offline phase before the parties communicate. It thereby provides a speedup of 4.5× compared to DeepSecure.

The XONN framework [112] creates secure inference through a novel protocol for conditional addition based on **Oblivious Transfer (OT)**. For OT, XNOR operations are of low computational cost [108]. Therefore, to create a more efficient setup, XONN replaces costly multiplication steps in the neural network with XNOR operations. Next to this, it also implements a trimming technique for neurons that contribute little to the inference. This adds up to an increase in efficiency compared to previously developed secure inference methods, (2.7× speedup on Cifar-10 over Gazelle [66], mentioned in Section 4.3). XONN is particularly efficient for larger CNNs, compared to previous studies.

Another advantage of using MPC methods, compared to homomorphic encryption, is that they can more easily also incorporate data verifiability, instead of only data confidentiality. An example of this is PVDLI by Reference [37], which applies secret sharing. Verifiability of the inference result of the cloud was previously the focus of Reference [50]. They did not use SPMC methods but instead made the server provide a simple short mathematical proof of the correctness of inference tasks using a sum-check protocol.

Reference [37] takes a different approach: They inject labeled verifiable data into the model input, which offers verification of the correctness of the inference results with a high probability.

Moreover, the secret sharing allows the injected verification data to be indistinguishable from the normal ones, so the malicious party cannot determine which are the verification data.

4.1.3 Secure Inference with Hybrid Methods. Several papers aimed to combine the best features of both SPMC and HE. As previously mentioned, one way to achieve this is to implement a HE method but replace the activation function with an MPC protocol instead of a mathematical approximation. This way, the most costly calculation of HE can be replaced with a more efficient and possibly more accurate protocol. Reference [81] was the first to attempt a hybrid setup by introducing **oblivious neural networks (ONNs)** with the *MiniONN* framework. ONNs are a combination of leveled FHE (using the now-outdated YASHE framework) and **oblivious transfer (OT)**. It uses the oblivious transfer to replace the non-linear functions (such as ReLu) with oblivious transfer protocols. It makes the basis of the non-linear functions into dot triplet products (see Appendix A.5). These are similar to the multiplication triplets in oblivious transfer, and therefore OT can be applied. This manages to highly outperform CryptoNets in terms of efficiency, especially for small numbers of requests that require quick responses.

Gazelle [66] continues the evolvement of facilitating secure inference using leveled FHE encryption, but, different from miniONN, it handles the non-linear functions through the application of GC. The goal of the paper is to use techniques that involve division-free arithmetic, as when they are converted to homomorphic operations, they are significantly slower compared to the operations on plaintext. Moreover, it uses the BFV scheme, instead of the YASHE in *MiniONN*. This method was further optimized by *Cheetah* [110], which focused exclusively on optimizing the process as proposed by Gazelle. It did not necessarily achieve speedup through a different cryptographic protocol but instead focused on speedup by optimizing the server side of secure inference through tuning and scheduling. This leads to a speedup of up to 79× compared to *Gazelle*.

Delphi [92] uses a (lightweight) cryptographic protocol, in combination with a planner, to speed up the architecture as developed by Gazelle. It uses a truncation trick to avoid overflow when multiplying secret-shared fixed-point numbers. Truncation limits the number of digits behind the decimal point. However, ABY3, discussed in Section 4.3, demonstrated that this trick can introduce large errors with non-negligible probability. This could result in erroneous classifications, so to verify that DELPHI avoids such errors, it was compared to miniONN, with a 99.2% overlap in prediction outcome, showing largely avoidance of these errors.

4.2 Secure Training

The previous section focused on secure inference, a common goal for distributed networks. However, there is also interest in training the network over distributed data, instead of using an already trained model. This is referred to as **secure training (ST)**. Usually, due to the nature of neural networks, if training is possible, then so is inference. Training implies forward and backward Many methods are adapted from the secure inference, and therefore also use HE, SMPC, or a combination of both. We will discuss these papers along the same axis, dividing between homomorphic methods and secure multiparty computation methods. The papers discussed are summarized in Table 3.

SecureML [94] provides the very first method that focuses on training as well as inference. They use the ABY framework [29]. *SecureML* proposes a new approximate fixed point multiplication protocol that avoids binary circuits and truncates decimal numbers and uses this to train neural network models. This technique is limited to two parties. *SecureML* provides both options for MPC and HE using, respectively, OT and LHE. However, the accuracy with LHE is very low, as higher degree polynomials could not be used, as they introduced too much overhead. For the OT solution, even by focusing on simple MLPs, the training time was in the thousands of seconds per iteration,

Table 3. Summary of Secure Training Papers

Ref.	Framework	Enc. Mechanism	Threat Level	No. servers	Collusion	Type of NN	Evaluation Datasets	Acc. on MNIST(%)	Efficiency	Fairness	Novelty
[94]	SecureML	LHE or OT	HBC	2	1 server + subset of clients	MLP	MNIST	93.1	High	–	introduces secure training with high computational cost
[93]	ABY3	OT GC aSS	MA	3	1 server + subset of clients	MLP	MNIST	94%	55,000× to [94]	–	adaptation of ABY to three-server setup
[61]	CryptoDL2	LHE	HBC	1	–	MLP CNN	MNIST a.o.	95	improves [94]	–	LHE for secure training, without re-encryption due to high cost
[156]	GeluNet	AHE (Paillier)	HBC	2	ND	MLP CNN	MNIST a.o.	96.9	0.281 s	–	AHE with the Paillier, leaving local calculations unencrypted
[136]	SecureNN	OT GC aSS	HBC MA	3	1 server + subset of clients	CNN DNN	MNIST	99	upto 93× to[94]	–	3-party setup for secure training, which can also protect against a malicious adversary
[2]	Quotient	OT	HBC	2	ND	MLP CNN	MNIST and others	99.4	50× [94]	–	ST with ternarized neural networks
[75]	MSCryptoNet	Multi-scheme FHE	HBC	1	–	CNN	MNIST Diabetes a.o.	99.6	*	–	Enables training phase using a multi-scheme FHE setup
[96]	–	FHE (HElib)	HBC	1	–	CNN	MNIST	96	40 min per batch	–	Homomorphic encryption for training
[28]	Fan4	SS	MA	4	Honest Majority	CNN (ResNet a.o.)	MNIST ImageNet	98.6	20× to [72]	Y	Introduces robustness while retaining privacy.
[102]	Blaze	GC SS	MA	3	1 server + subset of clients	DNN	Parkinson[118]	–	278× to [93]	Y	Introduce a new secret-sharing semantics for 3PC tolerating up to one malicious corruption
[137]	FALCON	SS	MA	3	Honest Majority	MLP CNN	MNIST CIFAR10 T-ImageNet	98.6	6× to [136]	–	Batch normalization in a 3PC MA setting
[109]	Trident	GC SS ABY	MA	4	ND	MLP CNN	MNIST a.o.	98.3	187× to ABY3	Y	Fairness for 4PC MA scenario
[72]	SWIFT	ABY	MA	4	Honest Majority	MLP CNN	MNIST CIFAR10	*	*	Y	Robustness, guaranteed output delivery, for HBC
[117]	ARIANN	FSS	HBC	2	1 server + subset of clients	MLP CNN	MNIST CIFAR10 T-ImageNet	99.2	42 h	–	Offers ST suitable for GPU
[73]	Tetrad	ABY	MA	4	Honest Majority	MLP CNN	MNIST CIFAR10	ND	4× to [109]	Y	Robust ST for MA scenario

The papers are sorted by year of publication and by alphabetical order of the author. Efficiency: Speedup compared to benchmark, complexity, or seconds, in most ideal scenario. Collusion and Drop Out: max. number of participants. Fairness: whether the framework offers at least fairness.
Abbreviations: FSS = Function secret sharing GMW= Goldreich-Micali-Wigderson, LHE = Linear Homomorphic Encryption, LSTM= Long short-term memory, MA= malicious adversary, ND= not determined.

and therefore too long and not practical. The works in the following sections try to improve on this work, either through the use of homomorphic encryption or secure multiparty computation.

4.2.1 *Secure Training with Homomorphic Encryption.*

Following *SecureML, CryptoDL2* [61] was the first to focus on homomorphic encryption for training and inference. It implements FHE using the HElib library, and approximations of activation functions were based on Chebyshev polynomials [147]. In comparison to *secureM*L, it offers both a significant speedup and an increase in predictive performance. However, the approximated polynomial activation functions caused training to be unstable. To manage the computational complexity and noise, by avoiding the costly bootstrapping of homomorphic encryption, their approach allowed some communications between client and server when the noise reaches a threshold. While this increases communication overhead, it is still lower compared to SPMC protocols, where several communications are necessary for each operation.

Nandakumar et al. [96] focus on training a network with FHE fully in the fixed-point domain and achieve convergence and reasonable accuracy. They argue that for other papers that only employ floating point operations for some intermediate computations and truncate them, errors can quickly accumulate and prevent convergence, especially in a non-interactive protocol such as FHE. Reference [96] achieves FHE in the fixed-point domain with bootstrapping by pre-computing in a table and performing homomorphic table lookup. This optimization leads to a 50× speedup compared to their naive setup, with 96% accuracy.

GELUnet [156] solves the issue of high computational complexity in approximations of activation functions differently. They use partial homomorphic encryption with the Paillier system but leave the local calculations unencrypted. In this setup, the input for the activation (for example, the encrypted intermediate weighted sum) is sent back to the client. The client has the key and thus decrypts the input executes the activation, re-encrypts the result, and sends it to the server for the next layer. While this limits computational cost, it leads to higher communication and higher necessity of compute power of the client side. As a main reason for ST can be to avoid this, Reference [156] might not be suitable in many situations.

MScryptonet [75] aims to provide an architecture for the secure training of the model through the application of a multi-key fully homomorphic encryption architecture. Previous frameworks, such as *CryptoDL2* [61], raise the option of HE model training, but with the encryption of only one key, thereby safely allowing only one client at a time. Thereby encryption is outsourced, but not necessarily collaborative among multiple parties. Multikey FHE can be a solution to these issues. A **multi-key fully homomorphic encryption (MK-FHE)** scheme allows a server to evaluate circuits over ciphertexts encrypted under different keys (see Appendix A.4.4). To achieve ST, *MScryptonet* uses approximations of activation functions with low-degree polynomials. Although training time was not reported, there was a 2.5× speedup compared to *CryptoNets* for inference. While multikey encryption solves the problem of collaboration among clients, another approach is applying SPMC, as discussed in the next section.

4.2.2 Secure Training with ***Secure Multiparty Computation (SPMC)***. Most methods for secure training use MPC techniques. A first example of this is *ABY3* [93]. Their approach is based on the previously developed ABY framework [29] similar to *secureML*, but extended it for three parties with support security against malicious adversaries. In this mixed method framework, they introduce new approximate fixed-point multiplication protocols for shared decimal numbers. This provides a more workable framework: Next to better security, they report a speedup of 55,000× in the most extreme scenario compared to *SecureML*.

Concurrently, Wagh et al. [135] provides a secure training setup for CNNs. It provides three- and four-party secure computation protocols, with various neural network building blocks such as matrix multiplication, convolutions, ReLu, Maxpool, and normalization. Its predictive performance is good, providing a 99% accuracy with a speedup towards both secure inference and ST setups. The protocol is established in such a way that it also protects against attacks from a malicious adversary, where no single party learns any information about the data. By creating more efficient communication, and eliminating oblivious transfer protocols, it creates up to a 407× increase over *SecureML*.

Another approach is taken by Agrawal et al. [2], which focuses on neural networks with ternarized network weights, similar to FHE-DiNN [15] and *TAPAS* [119]. Therefore, ternary matrix-vector multiplication is a crucial primitive for training. They implemented a protocol for ternary matrix-vector multiplication, which is based on Correlated Oblivious Transfer [6] that combines Boolean-sharing and additive-sharing for efficiency. Compared to *secureML,* they obtain an improvement of 50× in WAN time and 6% in absolute accuracy while working in the honest but curious setting.

Falcon [137] provides a 3PC setup that gains efficiency by a factor of 2 by rebuilding a derivative of ReLu. Moreover, to create a better accuracy, they also enable batch normalization. It offers MA security: Falcon provides for secure training in an honest-majority adversarial setting. It is 4.4× faster than *ABY3* and 6× faster than *SecureNN*.

Blaze [102] takes an approach with three servers, tolerating up to one malicious corruption, using new secret-sharing semantics. They also introduce a dot product protocol, where communication is independent of vector size, and a less expensive truncation method than *ABY3*, leading to an increase of up to 278× for neural networks. *Swift* [72] extends Blaze to offer maliciously secure, **three-party computation (3PC)**. It is robust in the honest-majority setting and also offers maliciously security for 4PC. Moreover, it is as fast as Blaze.

Trident [109] and *Tetrad* [73] build on the methods from *Blaze* [102] and *SWIFT* [72]. *Trident* [109] uses 4PC and has an additional honest party in the protocol. This leads to a speedup of up to 45× compared to *ABY3* for CNNs. While *Trident* offers fairness, *Tetrad*'s [73] protocol also comes with robustness. They provide this robust option by figuring out the trusted party in the protocol and delegating the calculation to them. While this is secure in a cryptographic sense, it is not the privacy-preserving solution that would be preferable in distributed learning.

Fan4 [28] focuses on providing a robust 4-party system. They mention that previous frameworks provide their robustness by delegating the calculation to a trusted party. While this is secure in a formal sense, in a real-world scenario, most participants will not accept this as a suitable solution. *Fan4* [28] introduces robustness, as well as private robustness, where the protocol can continue privately. They achieve this in an actively secure 3PC protocol for one corruption. *AriaNN* [117] uses their previously developed method of function secret sharing [16]. This is different from classic secret sharing, where a shared input is applied to a public function. Function secret sharing applies a public input on a private shared function. While offering a relatively minor speedup to previous methods, their framework is implantable on a GPU.

4.3 Summary of Collaborative Learning

We can see both for secure training and secure inference, the increase in accuracy, efficiency, and security as methods have improved over time. Importantly, we also start to see protocols, especially for ST, that also take into account malicious adversaries. The two situations, SI and, ST, are highly related to each other, however, the ideal setup can differ between the two purposes.

For secure inference, the largest group of papers consists of FHE-based security. Those methods assume Honest-but-Curious clients and servers, though some, such as *XONN* [112], do remark on the possibility of adapting the framework to withstand a malicious adversary. For secure inference, a key question to raise is the extent to which a higher level of protection would be necessary. It is important to remember that most of the private training setups can also be used for private inference, so more secure and robust frameworks are available for secure inference.

With regards to secure training, implementing a situation protected against malicious adversaries seems more important, with a preference for MPC methods for private training. However, few papers have included differential privacy in their methods. Most papers only look at the safekeeping of the data and do not seek to combine this with verifiability. Especially secure training could be sensitive to a poisoning attack, which involves the manipulation of training data to inject false information into the model, which can hinder the training process. Incorporating mechanisms for verification and accountability would be a critical step toward enhancing the security of CBL systems. Moreover, it is worth noting that there are still issues regarding excessive computational times. While this is an active problem and efficiency is continuously improving, an alternative approach may involve distributed learning, as this can delegate computation to the participants, thereby potentially reducing computational time. This topic will be expanded on and discussed in the next section.

5 Distributed Learning

One of the most prominent methods within distributed learning is ***Federated Learning (FL)*** [91]. FL is a specialized form of distributed learning that enables multiple participants, each with their local data, to collaboratively train a shared model while keeping the data localized. Most of this chapter will focus on FL, which we will describe in the next section. However, It is important to note that several configurations of distributed learning do not follow this architecture, most notably *Split Learning* [134], where one model is split up among different parties. We will describe these methods in Section 5.4.

5.1 Description of a Federated Learning Workflow

5.1.1 FL Overview. FL involves iterative model training across decentralized clients (for thorough discussion, see Reference [69]). Clients train local models on their private datasets, transmitting updated parameters to a central server. The server aggregates these updates (e.g., via federated averaging) to produce a global model, subsequently redistributed to clients for further training. This process continues until convergence or a predefined performance criterion is met. This approach enables robust deep learning while preserving data privacy and mitigating centralization risks.

FL, while promoting distributed training without direct data sharing, does not inherently guarantee privacy. This misunderstanding stems from conflating data isolation with data privacy. Although raw data remains local, shared model updates (gradients, weights) implicitly reveal information about the training data vulnerable to inference attacks. Sophisticated adversaries can exploit these shared updates to infer properties about the original data, potentially leading to privacy breaches. The central aggregation process lacks intrinsic privacy mechanisms, creating further vulnerabilities to malicious actors or data interception. Repeated communication rounds exacerbate this risk.

It is crucial for our main discussion to distinguish between *cross-silo* and *cross-device* FL, as these variations impact the scale and nature of participation in the distributed learning process. In cross-silo FL, the setup typically involves fewer clients; however, these clients are usually entities that possess substantial datasets and computational resources. This contrasts sharply with cross-device FL, where a larger number of less reliable clients participate, but each typically has access to smaller datasets and has less computational powers. Cross-device FL scenarios often require tailored privacy-preserving algorithms to address the unique challenges presented by the scale and reliability of the devices involved, and the risk of data exposure or loss may be heightened due to the increased number of less secure endpoints.

5.1.2 Federated Learning Security and Privacy Workflow. Similar to the previous chapter, our focus remains on FL algorithms that are specifically designed for privacy preservation. We visualize the security and privacy mechanisms in the FL workflow in Figure 3. To address the inherent privacy vulnerabilities in FL, the integration of **differential privacy (DP)** provides a robust solution. The DP noise-adding mechanisms can be employed in different parts of the workflow. This will have implications for the performance of the pipeline, the privacy level, and the privacy surface of the model. To better understand this surface, we divide the differentially private federated deep learning methods into two broad domains: *local DP* and *global DP*.

Global differentially private federated learning In a **Global Differential Privacy (GDP)** scheme, a global, trusted curator receives non-perturbed parameters from involved parties in response

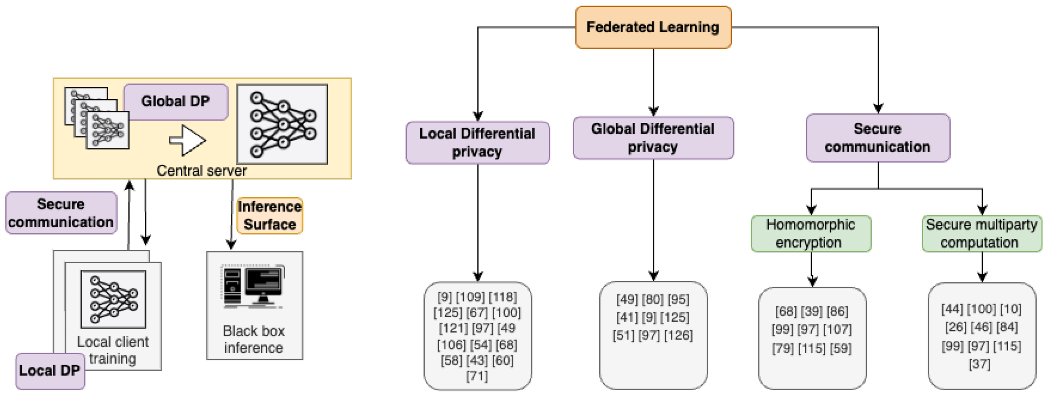

Fig. 3. An FL workflow demonstrating where privacy-preserving measures protect model inference and secure communication safeguards the training process (left). The distribution of research papers investigating these protective mechanisms is shown on the right.

to a query and applies random noise to the output of any queries being performed against the system. The randomized mechanism M is (ϵ, δ)-*differentially private* in the global domain if it satisfies Equation (1). *Local differentially private federated learning* In **Local Differential Privacy (LDP)**, each node applies its own set of noise to the network parameters, and what the curator receives is the perturbed data [14]. This ensures that if the orchestrator behavior is adversarial, then the amount of information leakage will be bounded by the DP guarantees of the system. The randomized mechanism M is (ϵ, δ)-*differentially private* in both local and global domains if it satisfies Equation (1).

Local and global privacy are added at different points in the FL workflow, as can be seen in Figure 3, but they are both based on similar noise-adding techniques. These techniques and their main challenges, such as the utility/privacy tradeoff, in FL are discussed in Section 5.2.

Beyond privacy, FL requires robust security measures. Cryptographic methods are employed to secure communication between the orchestrator and participants, protecting the integrity and confidentiality of both model updates and the aggregated model. Section 5.3 explores these security challenges, including scalability and verifiability. Section 5.2.3 covers approaches that combine noise-addition and cryptography to simultaneously address both privacy and security concerns.

5.2 Differential Privacy in Federated Learning

Noise-adding is the main method to achieve differential privacy in FL. The noise is usually added during the training to the internal state of the model, e.g., the model gradients. Considering the general workflow of FL with participants as the data holders and top and mid-level actors serving as orchestrators, the noise-adding mechanism can be employed in different parts of the workflow. The process of adding noise is carefully calibrated to ensure that while the individual data points are obscured the overall utility of the model is not significantly compromised. This balance is critical in maintaining both the performance of the model and the trust of the participants. A considerable number of proposed methods in the area of DPFL are application-oriented, as they trade the regular optimization method of their model with a DP version, usually DP-SGD, for local DP—e.g., References [9, 142, 151, 158]—or global DP—e.g., References [58, 106, 124]. The methods we cover in more detail in this section provide additional privacy and/or performance-related novelties. We summarize the selected differentially private FL workflow papers in Table 4.

Table 4. Overview of Private Distributed Learning Methods

Name	Privacy Level	PP Regime	PP Mechanism	Distributed Architecture	Type of NN	Evaluation Datasets	Highest Privacy Level
[124]	Global	(ϵ, δ)-DP	Laplace, Sparse Vector Technique	FL	MLP, CNN	MNIST, SVHN	$\epsilon = 1$ per parameter
[49]	Global	(ϵ, δ)-DP + MA	Gaussian	FL	N/D	MNIST	$\epsilon = 8$
[127]	Global, Local	(ϵ, δ)-DP	LDP [38]	FL	CNN, VGG	MNIST, FMNIST,CIFAR-10	$\epsilon = 5$
[159]	Global	f-DP	Gaussian	Private client models, shared noisy global model	CNN	MNIST, CIFAR-10	N/D
[59]	Global	(ϵ, δ)-DP + noisy gradients	Gaussian	FL	N/D	N/D	N/D
[122]	Local	(ϵ, δ)-DP	Gaussian	FL over wireless	N/D	N/D	N/D
[9]	Local	(ϵ, δ)-DP	Gaussian	FL cyclic	CNN	eICU, TCGA	$\epsilon = 3.84$
[58]	Local	(ϵ, δ)-DP	Laplace	FL	N/D	MNIST	$\epsilon = 1$
[132]	Local	(ϵ, δ)-DP	Gaussian	FL	SVM, DT, CNN	MNIST, UCI Nursery	$\epsilon = 0.5$
[154]	Local	(ϵ)-DP	Laplace	FL 3-tier	CNN	MNIST	N/D
[64]	Local	(ϵ, δ)-DP + MA	Gaussian with adaptive variance	FL + ADMM	Logistic Regression	Adult	$\epsilon = 1.01$
[82]	Local	(ϵ)-DP	Laplace, Randomized response	FL	N/D	MNIST	$\epsilon = 8$
[139]	Local	(ϵ, δ)-DP	Randomized response prior	FL	LDA	Amazon reviews, spam emails, sentiment	$\epsilon = 7.5$
[51]	Local	Renyi-DP + MA	Dirichlet	Ensemble	Teacher-student reinforcement learning	AirRaid-ram-v0 simulation from OpenAI Gym	$\epsilon = 4.46$
[68]	Local	(ϵ, δ)-DP	Gaussian	FL	CNN, LSTM	MNIST, EMNIST, CIFAR-19, Stackoverflow	$\epsilon = 19$
[70]	Local	(ϵ, δ)-DP + proposed accountant	Gaussian	FL	N/D	N/D	N/D
[89]	Local	(ϵ, δ)-DP + MA	Gaussian	FL	SqueezeNet	Diabetic retinopathy	$\epsilon = 11$

The papers are sorted by year of publication and by alphabetical order of the author. Privacy level: Reviewed works rarely share the same settings for different variables in these types of methods, e.g., model architecture, number of clients, rounds of training, and so on. The reported values here are the most significant and/or relevant results from the most common dataset used, usually MNIST if available. The privacy levels should not be a basis of comparison between methods and each method should be evaluated individually.
Abbreviations: CNN= Convolutional neural network, MA= Moments accountant, FL= Federated learning.

5.2.1 Private Training and Inference in Federated Learning. Geyer et al. [49] propose a global-DP-based FL mechanism based on adding noise to the communicated gradients of each client by the orchestrator. They use a random subsampling mechanism to pick K out of N participants for the parameter aggregation phase. Then, the updated model parameters are only sent back to this subset of participants.

Beaulieu-Jones et al. [9] use a cyclic approach, where each Learning Party will train their model until they spend their privacy budget, then pass on the computed differentially private set of gradients to the next Learning Party.

Zhao et al. [158] communicate the network weights from Learning Parties to the parameter server and average them to obtain and send back the optimum global model parameters for the next training iteration. They use a publicly available dataset to validate the performance of each learning party's local model and only choose the best-performing ones to average over to mitigate the problem of unreliable partners with noisy and low-quality datasets. To make sure the learning parties cannot infer which are dropped and which are selected, the parameter server uses the Exponential Mechanism to protect the selection mechanism. The total privacy budget will be the largest of the two privacy budget values—on the learning party to perturb the objective function and on the Parameter Server to obfuscate the selection process.

Hartmann and West [59] follow the local DP FL scheme by adding uniform noise to the local gradients before submitting them to the parameters server. However, on the orchestrator level, the collective anonymized noise values will be subtracted from the cumulative gradients at each step to achieve higher utility.

Wang et al. [139] propose a federated DP **Latent Dirichlet Analysis (LDA)** mechanism for privacy-preserving distributed text mining. The orchestrator is assumed to be untrustworthy. They propose a modified version of Warner's randomized response method called ***random response with priori (RRP)*** to make their federated LDA mechanism DP on local nodes. The orchestrator will attempt to infer the topic-word distribution from DP local updates received from each participant. The RRP method non-uniformly samples the noisy terms *a priori* drawn from the local model.

Seif et al. [122] consider an FL system connected over a wireless channel and propose leveraging the superposition characteristics of a wireless channel to preserve the privacy of the communicated gradients. They make use of one flat-fading Gaussian **Multiple Access Channel (MAC)** bound by Local DP constraints. The channel coefficient and additive Gaussian noise determine the amount of perturbation applied to the gradient sets. They provide tighter privacy bounds dependent on the number of clients on the wireless channel through their proposed transmission scheme.

Huang et al. [64] use the **Alternating Direction Method of Multipliers (ADMM)** to perform their distributed learning task and augment it with differential privacy. The ADMM method optimizes a Lagrangian function associated with the global loss function by alternately minimizing it with regard to local models and the global models. Based on the intuition that it is not necessary to solve the local optimization problems to a very high precision to guarantee global convergence, the Lagrangian function is replaced by a first-order approximation of itself. The authors use the Gaussian noise-adding mechanism to achieve DP and vary the variance of the noise with time, decreasing its magnitude as the number of iterations increases.

Kairouz et al. [68] address the fact that most DP-SGD-based methods rely on privacy amplification through subsampling by carefully choosing mini-batches, which can be challenging in a distributed setting. They propose the **follow-the-regularized-leader (FTRL)**, a DP-distributed training algorithm that does not rely on subsampling. FTRL adds correlated noise to the gradients, as opposed to the DP-SGD algorithm, in which independent noise is added to the gradient vector.

Although most differentially private distributed deep learning methods are in the context of generative and discriminative models, other types of deep learning methods can also benefit from this paradigm. An example is the work by Gohari et al. [51] on differentially private reinforcement learning. This work considers a deep reinforcement learning setting in which agents would ask for a demonstration from other agents to improve their performance following a teacher-student scheme. In the case where the private dataset of the agent contains sensitive information, the demonstration opens up the teacher to privacy attacks such as membership inference. They limit the context of demonstration to the teacher's policy and use the Dirichlet mechanism to privatize the discrete-space policy of the teacher.

5.2.2 Reducing the Impact of Privacy/Utility Tradeoff. The balance between privacy protection and data utility is a fundamental challenge in the design of differential privacy mechanisms. While stringent privacy guarantees often necessitate the introduction of noise or other obfuscating alterations to the data, these modifications can detrimentally affect the utility or accuracy of the resulting models. In distributed learning systems, this tradeoff becomes even more critical, as the system must maintain high data utility to ensure effective learning and decision-making while also upholding rigorous privacy standards. Addressing this challenge requires innovative approaches that optimize noise addition and data processing techniques, thereby minimizing the loss of utility without compromising the privacy of individuals. This section explores various strategies and methodologies developed to improve this delicate balance, aiming to achieve optimal outcomes in both privacy preservation and the practical usefulness of data.

Shokri and Shmatikov [124] in their seminal paper use the Laplacian mechanism of differential privacy to protect the communicated gradients. But, first, they select only a subset of the most prominent gradient values for communication and the Laplacian noise (generated according to a fixed budget) will only be added to this subset of gradients before communicating them to the parameter server.

Liu et al. [83] propose a DP FL scheme with a dynamic noise-adding mechanism for improved utility. They employ an extended version of the *layer-wise relevance propagation* algorithm and add noise calibrated to the calculated relevance of each attribute class. Furthermore, they propose a randomized privacy adjustment mechanism in which a threshold for the efficacy of each feature is defined by the user. The threshold is then used in a method similar to Warner's randomized response to decide whether and with what probability to add noise to a feature or just pass the data unperturbed.

Liu et al. [82] adapt the amount of noise they add to the data by using a modified relevance propagation algorithm to measure the contribution of each neuron of the input layer and scale the noise based on its impact on the model output. They further threshold the contribution of each feature with a parameter f. Features with contributions above the threshold will have noise injected into them, while for the nodes with contributions lower than the threshold, a randomized response mechanism is used to decide whether noise is added to the data or the original data passes through. Noise sampled from a Laplace distribution is added to the loss function.

Zhang et al. [154] design a three-tier FL mechanism to perform deep learning on resource-constrained edge devices. They pre-train the model on public data and then partition the model on the last layer among edge nodes to reduce their computational overhead. They apply local DP to the model on the client side.

Kim et al. [70] use an enhanced privacy budget accountant in a federated DP-SGD scheme and provide a variable sensitivity parameter—as opposed to a fixed bound on gradients—leading to tighter privacy bounds. Furthermore, they provide an analysis of the communication rate, privacy level, and model utility, which leads to a new way to calculate the DP noise variance, resulting in a lower transmission rate and higher utility while maintaining the same level of privacy.

5.2.3 Reinforcing Privacy while Maintaining Security. We earlier discussed the privacy and security risks differ by nature and require different treatments. DP by nature can protect the training and inference surfaces of a distributed learning system against specific privacy attacks. However, to mitigate security risks, existing research often combines private and secure training and inference methods. This could potentially complicate the dynamics of both the training phase and inference phase with regards to compute/time costs and the tradeoff between utility and security/privacy and needs careful consideration and development. One example is the work by Hao et al. [58], which combines local DP with homomorphic encryption to prevent malicious parties from modifying the communicated gradient values. They encrypt the DP gradients with PPDM additive homomorphic encryption method [160] to provide a more lightweight option. This allows for an encrypted addition of the model updates at the server level. They propose a scheme that is robust against participants dropping out and that preserves privacy against malicious Learning Parties. Phuong et al. [106] also use homomorphic encryption in addition to DP for private, secure training of a deep model.

Hartmann and West [59] imagine a situation where the parameter server is malicious and could collude with several Learning Parties. To mitigate privacy concerns, they add uniform noise to the local gradients and send them to the parameter server. Simultaneously, they send the actual noise vector through an anonymizing service like Tor so the server cannot recover the exact gradient values of any single Learning Party but would be able to subtract the sum of noise values from the sum of gradient values and compute the gradient values for the next step of the training process with high utility. Truex et al. [132] combine local DP with secure multiparty computation between participants to provide security during training. Using an adjustable security parameter, they are able to reduce the amount of calibrated noise added to the local model parameters during the training phase.

Hao et al. [58] apply both DP and homomorphic encryption to the FL protocol by using additional HE to provide a more lightweight option. It does this by applying ϵ-DP by the Laplace mechanism to the model updates and thereafter encrypting them in an AHE-applicable manner. This allows for an encrypted addition of the model updates at the server level. They propose a scheme that is robust against participants dropping out. This is expanded and improved on by Reference [128], which proposes a multi-key HE scheme; this method is secure when the server colludes with one of the participants.

Sun et al. [127] augment the local DP FL pipeline by masking the identity of each participant from the orchestrator through the utilization of techniques such as IP masking or secure multiparty computation. The weights of local models are split and shuffled before being received by the orchestrator. Each client samples a random time for sending its parameters to the server. Each set of weights is split and then the weights of different local sets are shuffled before being sent to the orchestrator. On the local level, the authors add noise to the gradients of the deep net using a randomized noise-adding mechanism to make sure the overall noise has a zero mean.

Zheng et al. [159] introduce a DP FL mechanism utilizing the notion of Gaussian f-DP [34]. They address two threat models, one assuming one malicious learning party and the other assuming several malicious parties, and introduce *weak* and *strong federated f-differential privacy* schemes to deal with these threat models, respectively. The general framework involves training customized models on each client—as opposed to federated averaging in which, at each round, clients adopt the global model—with the help of a noisy global helper model. At each round of training, the selected participants pull the helper model and utilize it to increase the efficiency of their private local model. The local DP model is then synchronized to the server to update the helper model.

Malekzadeh et al. [89] propose *Dopamine*, a federated differentially private deep learning mechanism for medical data processing. They employ DP-SGD for local training. To provide hospital-level privacy guarantees, they use a secure aggregation method based on homomorphic encryption that is performed on collected local noisy DP models.

The work of Hao et al. [58] is expanded and improved on by Reference [128], which proposes a multi-key HE scheme in a similar way. This way, the method is secure when the server colludes with one of the participants.

Ma et al. [88] also use multikey HE, specifically a multi-key protocol based on CKKS. There, the model updates are encrypted via an aggregated public key before sharing with a server for aggregation. Collaboration between all participating devices is required for decryption. It is secure for collusion until n-1 participants.

5.3 Security Methods for FL

Not only differential privacy mechanisms are important for safe distributed learning, but security measures can also be incorporated to improve data and model privacy. The papers discussed in this section are summarized in Table 5.

Table 5. Encryption-based Protective Measures for FL, Sorted by Year and Author

Ref.	Framework Name	Encryption method	Threat level	Collusion	Type of NN	Evaluation Datasets	Acc. on MNIST	Efficiency	Drop Out	DP	Novelty
[13]	SecAgg	SS	HBC	$n/3$-1 participants	–	–	–	O(N^2)	$n/3$-1	Gaussian	Introduces secure aggregation of models in FL
[105]	PPDL	AHE (LWE, Paillier)	HBC	No collusion	MLP CNN	MNIST SVHN [97] a.o.	99	2.40× loss to PT	None	Laplace	AHE-based scheme for cross-silo FL
[58]	–	AHE (PPDM)	HBC	Multiple participants	CNN	MNIST	98.1	100× to [105]	Arbitrary subset	Laplace	Combines AHE and DP for efficient FL
[128]	–	AHE Re-encryption	HBC	Any participant	ASGD NN	MNIST	97	2.64× loss to [105]	None	–	Multikey setup using re-encryption and ASGD
[146]	Hybrid-Alpha	FE	HBC	Maj. honest	CNN	MNIST	–	68% to [132]	Dep. on collusion	Gaussian	Functional encryption scheme resistant to dropout and addition of participants
[10]	–	SS	HBC SMA*	5%	–	–	–	–	5%	Shuffle [8]	Allows for secure aggregation with DP shuffling
[36]	SSDDL	SS	HBC	Some participants	CNN MLP	MNIST SVHN	99	O(nN)	ND	–	Effective secret sharing that can handle collusion
[67]	FastSecAgg	SS: FastShare	HBC	10%	CNN	–	–	–	10%	–	Secret sharing scheme based on Fast Fourier Transform
[125]	TurboAgg	aSS	HBC	Maj. honest	CNN	CIFAR10	–	O($N \log N$)	50%	–	Multi-group circular strategy for efficient model aggregation
[145]	PPFDL	GC Paillier	HBC	No collusion	CNN	MNIST	95**	Improves [13]	Arbitrary subset	–	Solution to reduce the negative impact of irregular users on accuracy
[143]	VerifyNet	SS HE hash	HBC	(t - 1) participants***	CNN	MNIST	99	High overhead	30% tested	–	Double-masking protocol to encrypt users' local gradients and provide verification
[153]	BatchCrypt	AHE (Paillier)	HBC	No collusion	MLP CNN LTSM	FMNIST SSP [130] CIFAR10	–	93× to FATE [84]	None	–	Uses batch-encrypting for more efficient cross-silo FL
[120]	Poseidon	FHE (CKKS)	HBC	(n - 1) participants	MLP CNN	MNIST CIFAR100 o.a.	89.9	5,283.1 s	None	–	Multiparty HE for unlimited amount of participants
[162]	DAEQ-FL	ElGamal SS	HBC	Maj.honest	CNN LSTM	MNIST CIFAR10 SSP	99	****	t***	–	FL with no trusted third party and non-i.i.d. data
[57]	–	SS	HBC	(t - 1) participants	–	–	–	–	ND	–	Extension of [143] for non–fully trusted centers
[88]	xMK-CKKS	MK-CKKS AHE (Paillier)	HBC	(n - 1) participants	CNN	UP-FALL [90]	–	****	ND	–	Multi-key setup efficient, requiring all parties for decryption

Efficiency in speedup compared to benchmark, complexity, or seconds. Collusion and Drop Out quantified in max. number of participants. Abbreviations: DM = double masking, ND= not determined, SSP = Shakespeare, PT= plaintext
* SMA = semi malicious ** with irregular users *** in (t-1), t refers to t-out-of-n SS **** papers show improvement compared to a similar Paillier-based protocol but do not compare to a benchmark.

5.3.1 *Security of the Updates.*

Phong et al. [105] showed that in the DP setup of Reference [124], the shared gradients would still cause severe information leakage of the local data. To prevent this, they proposed AHE with the Paillier system and **learning with errors (LWE)** [79] to encrypt the shared gradients. They specifically consider the concept of a dishonest server. It leads to higher security, but it does not consider the dropout of participants.

HybridAlpha [146] takes a different approach by using functional encryption as a security measure. For functional encryption systems, the decryption key does not allow the user to learn the encrypted data, but instead only a function of the encrypted data (see Appendix A.5.4). *HybridAlpha* uses this to provide security with a dishonest client and an HBC server. Moreover, *HybridAlpha* combines this with ϵ-DP, added by the participating parties. *BatchCrypt* [153] proposes an AHE scheme for encoding of batch gradients, as batched training might lead to better predictive performance. It does not increase the communication costs but does lead to an increased computational cost for the clients. *SSDDL* [36] uses secret sharing, but instead of sharing gradients, the clients split their local gradients into shares and distribute them to each other. When they receive the shares, the participant calculates aggregation results and uploads that to the server, which updates the global network parameters. It is secure for an HBC server colluding with HBC participants. *DAEQ-FL* [162] uses Paillie with efficient threshold encryption. This leads to good predictive performance and lower computation and communication costs. Moreover, it works with no trusted third party and can handle non i.i.d. data.

5.3.2 Security of the Aggregated Results. While most papers focus on the clients' gradients, some also argue for the security of the final model weights. *PPFDL* [145] created a protocol based on the Paillier Scheme and Garbled Circuits. They incorporate a method to reduce the impact of irregular clients on the accuracy. While increasing the server's computational cost, the clients' computational and communication costs are significantly lower than Reference [13]. The scheme also protects the aggregated result. *Poseidon* [120] uses multiparty homomorphic encryption (CKKS scheme) with distributed bootstrapping. It focuses on protecting both the model updates as well as the final model weights. It functions under an HBC model with n-1 parties colluding. Moreover, it argues to be the first framework that allows for quantum-resistant distributed learning.

5.3.3 Security of Cross-device FL. Most of the papers in this chapter focus on cross-silo FL, where a few clients with larger amounts of data participate in FL. The alternative is cross-device FL: many clients with smaller amounts of data, for example, mobile devices. For cross-device FL, it is essential to handle large groups of participants and be robust to clients dropping out. Secure Aggregation [13] proposed the first approach to FL that is both privacy-preserving and robust to users dropping out. It privately shares the gradients between the clients and the server using additive secret sharing: Each client double masks the gradient and uploads that to the server. Sharmir's t-out-of-n secret sharing scheme is adopted to handle the dropout of participants. When receiving enough masked gradients, the server aggregates them and the pairwise masks cancel out. They prove that even when the server and t-1 clients are colluding, no client information is leaked. However, while the pairwise structure is secure and robust to dropout, the overhead grows quadratically with the number of clients, which limits its application to large-scale scenarios. Several frameworks have aimed to improve the overhead and speed up aggregation.

Bell et al. [10] continue on secure aggregation by proposing a protocol that allows for (poly)logarithmic overhead. They argue that every client double masking with all the other clients is not necessary, and they propose a more efficient communication graph. Their HBC secure aggregation claims to handle a billion clients, and their more secure semi-malicious setup can handle thousands of clients. Their scheme can handle up to 5% of corrupt clients and 5% of dropouts and allows for differential privacy through secure shuffling [8]. FastSecAgg [67] focuses on speedup of the aggregation by developing a novel multi-secret sharing scheme, FastShare, which is based on the **Fast Fourier Transform (FFT)**. It leads to computational speedup with similar communication and protects against adaptive adversaries. FastSecAgg does struggle with handling dropout of clients (10%). TurboAgg [125] limits the overhead to O($NlogN$) from O(N^2) and is robust to a 50% dropout. It limits overhead by performing the SS in groups. The users are divided into several groups, and at each aggregation stage, the users in one group pass the aggregated models of all the users in the previous groups and the current group to users in the next group. The running time is almost linear with the number of clients.

5.3.4 Verifiability. VerifyNet [144] broadens the scope of privacy protection by focusing on verifiability next to confidentiality. This is achieved by exploiting a homomorphic hash function into the underlying structure of VerifyNet with a double-masking protocol to guarantee the confidentiality of users' local gradients during the FL. It does not only consider the confidentiality of the model updates to the server, but also the integrity of the returned average. As such, it offers protection against a malicious server. However, the double-masking protocol is computationally expensive, and a trusted third party is still needed to establish keys. Reference [57] is an extension of VerifyNet: They also use a double-masking protocol but propose a secure scheme where the key generation center is HBC.

5.4 Other Distributed Learning Architectures

5.4.1 Split Learning. **Split learning (SL)** [134] is another method that falls under distributed learning. With SL, a deep neural network is split into multiple sections of several layers, each of which is trained by different parties. Generally, this process works by transferring the weights of the last layer of each part to the next in forward propagation, without sharing any raw data—only the weights of the last layer (the cut layer) are sent to the next client. Afterwards in the backpropagation, the gradients will be sent back: from the first layer of the second party to the last layer of the first party. SL allows for different configurations, including vertical SL, where the first layers are vertically split for vertically partitioned data, or the last layers are trained by the party with the labels. It also allows for a combined federated and split network, where the earlier layers are trained in a federated manner [131].

While SL offers promising new methods of handling distributed data and models, it comes with several security and privacy issues, due to the sharing of the cut layer during forward and backward propagation. The different possibilities in distribution can also lead to different privacy issues. Reference [101] highlights the necessity for extra security or privacy on SL by showing its susceptibility to inference attacks both from malicious servers and malicious clients. They also showed that previously proposed distance correlations [133], which showed more privacy to reverse engineering attacks than DP-SGD, are not secure. Moreover, Reference [42] highlights how even an HBC server can infer the data from participants, especially if the NN does not have many layers in depth. Recently, there have been several studies that offer potential solutions to these insecurities in SL. Reference [78] highlights another issue: label leakage. In a setup of one party having the features and the other party the data, the second party can recover the labels during the training process. This paper protects the confidentiality of the label by adding noise perturbation that is optimized to prevent label leakage. Reference [104] offers a solution based on HE with a CKKS scheme for a single client and server scenario. It encrypts the client's cut layer gradients that serve as input for the server layers.

5.4.2 Ensemble Methods. Ensemble methods are a fundamental technique in machine learning that involves combining the predictions from multiple models to improve the robustness, accuracy, and generalizability of predictions compared to any single model alone. These methods are based on the principle that a group of "weak learners" can come together to form a "strong learner." Ensemble methods and FL both involve training multiple models on distributed datasets, but they serve different primary purposes. FL focuses on training models collaboratively across multiple decentralized devices or servers (clients) holding local data without sharing that data centrally, thereby preserving privacy and reducing data centralization. This framework can naturally incorporate ensemble methods by using the local models trained on each client as members of an ensemble, where their predictions can be aggregated to improve overall model accuracy, robustness, and generalization. In the context of distributed differentially private deep learning, ensemble methods can be particularly beneficial, as they can enhance model performance without compromising the stringent privacy requirements. For example, each client can add noise to their local model's outputs (or during the training process) to meet differential privacy criteria, and the central server can then aggregate these differentially private models, potentially using techniques such as weighted voting or averaging. This aggregation can help in mitigating the noise added for privacy, leading to both a robust ensemble and adherence to privacy guarantees, effectively balancing utility and privacy in distributed environments.

In the context of DP ensemble methods, one of the most prominent examples is the **Private Aggregate of Ensemble Teachers (PATE)** framework. Papernot et al. [99] proposed this framework as a DP learning method where multiple teacher models, each trained on disjoint subsets of sensitive data, contribute to the training of a student model without directly exposing the sensitive data. The PATE approach aggregates the outputs of the teacher models such that the privacy of the underlying data is preserved, typically using mechanisms like noisy voting. Several variants enhance the robustness and privacy guarantees of the PATE framework. For instance, Papernot et al. [100] introduce a scalable version of PATE with granular control over the noise added to the teacher's outputs. The aggregation of teacher responses is followed by the addition of noise calibrated to the sensitivity of the counting queries, which are indicative of the number of teachers agreeing on a particular label. This noise addition is governed by the Gaussian or Laplacian mechanisms, aligning with DP standards. It also addresses the challenge of scaling PATE to a larger number of teacher models and a broader set of data classes. Structuring the teacher ensemble into several smaller groups and performing noisy aggregation within each group before a final aggregation step reduces the overall noise required for a desired privacy guarantee, which could improve the utility of the student model. PATE framework has been since adopted for many different use cases, e.g., G-PATE by Long et al. [86], which adapts the core principles of PATE for **generative adversarial networks (GANs)**, enabling the training of DP data generators that can produce high-quality synthetic data while preserving the privacy of original dataset with private aggregation happening among different discriminators.

5.5 Summary of Secure and Private Federated Learning

Due to vulnerabilities of federated and other distributed learning methods to information and model leaks, privacy-preserving methods to mitigate these problems are being increasingly developed and advanced. The majority of methods for privacy preservation in distributed learning in literature are differentially private methods. Following the proposal of the DP-SGD method [1], many deep learning methods have adopted this DP optimization method to perform FL in a privacy-preserving manner. The selection of the DP regime is usually problem-dependent and impacts both the utility of the model and the "measured" privacy budget spent. The threat model also has an important role in major decisions like whether to implement global or local DP. However, these choices in the majority of the existing works in literature do not impact the privacy method itself—DP-SGD. The methods we focused on in this section push the private training and inference of distributed DL model further. The main motivation for the reviewed DP methods can be categorized into *improving the privacy/utility tradeoff* and *improving the security as well as the privacy aspects.* Since adding privacy to DDL models generally comes at the cost of utility, the methods focused on this aspect try to improve the utility of the model while maintaining an acceptable level of privacy. On the hybrid approaches, the researchers recognize that in some use cases the privacy itself is not enough and based on the assumptions about the actors—or lack thereof—and the setup of the training and inference pipeline, we need to combine security and privacy methods such that the method would provide acceptable privacy and security, high utility, and a seamless combination of these two aspects.

With regard to secure communication in FL, we notice a difference between cross-silo and cross-device FL. Most DP methods focus on cross-silo FL. HE seems to have the preference in cross-silo situations, while SPMC and especially secure aggregation based on SS are often applied to cross-device settings. This might be related to the scalability of the different methods. Different from the CBL frameworks in Section 4, we see fewer papers that are resistant to a malicious server; only Reference [10] considers a semi-malicious scenario. In Section 5.2.3, we discuss the methods that combine security and privacy. It is an essential avenue of research that aims to offer solutions for both security and privacy threats. Many of these methods either offer improvement on the privacy or the security side of the mechanisms. It is important that these papers properly investigate the issues that come both with differential privacy and cryptographic methods.

We also regarded two other distributed learning methods: **split learning (SL)** and ensemble methods. SL offers an interesting new avenue in distributed methodologies. Not only could it offload model computation to different parties, but it also shows opportunities for handling vertically distributed data, which is often more complicated and understudied. However, SL still has significant security and privacy issues that need addressing. Ensemble methods are an interesting concept that could potentially help reduce the impact of noise-adding for differentially private FL. Due to the configurations of the framework, it is also scalable and offers opportunities for GANs. However, an evaluation of robustness in security would be beneficial.

6 Discussion

6.1 Design Tradeoffs for PSDDL

When designing a PSDDL system, several tradeoffs can be made, as illustrated throughout this article. There is not a single best solution that addresses the privacy and security needs for each possible type of distributed workflow. Sections 4.4 and 5.4 provide a summary of how workflow impacts the protective mechanism choice. In this section, we highlight several design choices that need to be considered when deciding on PSDDL methods. We highlight tradeoffs of security and privacy methods and differences between DDL and CBL. Other factors that play a role are how the DP is accounted, and the threat level and model.

6.1.1 Security/Privacy and Accuracy/Efficiency Tradeoffs. Several papers report an inverse relationship between enhanced privacy measures and operational efficiency (or accuracy) of the system. This inverse relationship necessitates a detailed articulation of system requirements, prioritizing either privacy or efficiency based on the specific needs of the application. The implementation of DP methods further demonstrates the tradeoff between privacy and accuracy. DP is a technique designed to ensure that the output of a database query is minimally affected by any single individual's data, thereby offering strong privacy guarantees. However, as privacy constraints are tightened (by reducing the probability of an individual's data influencing the output), the accuracy of the data output can diminish. This reduction in accuracy occurs because more noise is introduced to mask individual contributions, which can obscure meaningful insights.

Moreover, the deployment of advanced security methods often results in decreased system efficiency. Even with modern advancements in cryptographic techniques and secure computation protocols described in Section 4 and Section 5, we still see a significant computational overhead introduced by these security measures compared to unencrypted computations. While the direction of research should, and will, focus on more efficient protocols, it is important to highlight this payoff. Deciding on the level of security requires a careful look at what the application needs and what risks it faces. For example, while the strong security frameworks from secure training can be useful in secure inference, the high computational cost is an important consideration. Evaluating the necessity of advanced security measures can help find a balance between maintaining security and ensuring the system works efficiently. Similarly, using both DP and ENC can greatly improve accuracy and security. However, it is important to think about whether such high levels of security are always necessary. For example, if the task involves only inference and not training, then simpler security measures might be sufficient. Decisions on using these advanced technologies should consider their impact on computational resources and whether they are essential for the specific application.

These points of payoff between security and privacy versus accuracy and efficiency lead us to an important issue in the field of PSDDL. Currently, there is a notable absence of a comprehensive framework to quantify the tradeoffs between privacy, accuracy, and efficiency. Such a framework would facilitate the evaluation of different security and privacy methods under varying operational constraints, workflows, and preferences.

6.1.2 Collaborative Learning vs. Federated Learning. One major challenge with CBL is speed. Although research efforts are continuously making these methods faster, the high computational demands can still be a significant issue. An alternative approach in that situation may better be found in an FL setup, as this can delegate computation to the participants, thereby potentially reducing computational time, especially if scaled in a cross-device scenario. Therefore, where computational power is available should be taken into account, combined with the loss in efficiency for all setups. Another issue that can affect the choice between CBL and FL is the choice of data and model ownership. In a collaborative situation, while secure, the data is still sent to the central server. Thus, the parties are not fully in control of what happens with their data. Similarly, in a situation where the central party wants to have control over or ownership of the model, it might not want to share it with FL clients. Whether due to risk minimization, business incentives, legal or ethical constraints, this should be considered in the multifaceted decision of selecting a PSDDL setup for a particular use case.

6.1.3 Differential Privacy Accounting. Privacy costs of noise-adding are incurred by the iterative running of the DL algorithm on the training dataset. Calculating the privacy loss of the DP algorithm during the training phase, however, is not a trivial task. Different "accounting" methods exist for the composition of DP guarantees of DP mechanisms. Usually, the accounting mechanism works under a specific DP regime, e.g., f-DP, and supports the application of certain DP mechanisms, e.g., adding calibrated noise from certain distributions. These accounting methods range from the application of composition theorems and advanced composition theorems of DP [39] to widely adopted moments accountant [1] to more recent methods such as Edgeworth Accountant [138]. The problem definition limits the choice of the regime, mechanism, and accounting method. Nevertheless, care needs to be applied in the selection of each aspect of the privacy-preserving mechanism to increase the chance of success in training a model with high levels of privacy.

6.1.4 Threat Models and Security LevelS. As discussed in Section 6.1.3, the problem definition limits the range of available choices while designing a DP algorithm. One of the problem aspects that can inform the problem definition is the security and threat parameters. Putting aside the black-box access level of the external party, assumptions about the participant and orchestrator—e.g., trusted, honest but curious-could impact the choice of privacy preservation mechanism and also the performance and privacy level of the outcome of the training as well as the training process itself. One of the main points to mention here is the issue with homomorphic encryption. Methods with HE, especially secure inference by HE, do not often contain proper security analyses. HE can be considered Honest-but-Curious by itself, but in real-life scenarios, the security requirement is often stricter. Moreover, there are also still the issues with regards to GDPR, as there is still discussion about whether protection by HE would be GDPR-compliant [121].

Finally, as pointed out in Reference [28], there can be a difference between a formal definition of privacy and how this would be regarded in practice. Privacy as defined by stopping information leakage by models can happen during two stages: training and inference. For inference privacy, since usually a central model trained in a distributed manner on several disparate datasets will be exposed to real-world use cases, taking measures such as making the model differentially private should protect the model. However, during the training phase, the malicious actors could be either the participants themselves or the orchestrator. Considering the flow of information in a typical distributed ML setup, i.e., partial updates shared by participants with the orchestrator and central updates shared by the orchestrator with the participants, to ensure privacy in a no-trust setup, we need to make the output of both the orchestrator and the participants private. In DP, this will be done through global and local DP methods, respectively.

6.2 Current Research Gaps of PSDDL

While the field of PSDDL has grown substantially in the past few years, there are still some issues that remain understudied. We have highlighted some of the current issues in PSDDL that are important for the safe, accurate, and fair application of PSDDL in practical scenarios. We will highlight the issues that we believe deserve more detailed studying,

6.2.1 Data Distribution and the Problem of Non-i.i.d Data. Since the focus of this article is on DDL methods, it would be beneficial to look at the assumptions the reviewed methods make about data distribution. Most assume independent and identically distributed data among different nodes. Furthermore, these methods usually assume i.i.d classes for classification tasks. While this would help in developing research, real-world use cases often do not benefit from such an advantage. On one hand, in many cases there exists some form of skewness and class imbalance in the data. It can be easily observed in the standard vision datasets such as CIFAR and MNIST, which are used by almost all of the reviewed methods and do not have any imbalance in their classes—digits 0–9 for MNIST, object classes for CIFAR. The situation gets more complicated with tabular data, where the dataset could easily have multiple categorical or mixed features. Even considering a "class" feature for the dataset, it still could be the case that there are under-represented classes in other categorical features, and while the method would potentially show high utility overall for predicting the "class," it would perform poorly for some of the under-represented classes.

The distributed nature of the methods we have reviewed adds another complication, which is the distribution of the data among participants. Regardless of the distribution of the data among classes in the original dataset, many real-world factors such as geographical location and temporal aspects of data gathering could impact the distribution of the data among nodes and lead to datasets of participants having distributions differing from the original distribution of the whole dataset. It is important to note that with collaborative learning, due to the setup of the server building a singular model, the effect would be less severe than with distributed learning methods such as FL.

6.2.2 Measuring and Validating Privacy Level. In this survey, we have reviewed a wide variety of protective mechanisms for distributed deep learning. Most of these methods employ DP and cryptographic methods, either separately or in combination. Each method provides its own guarantee of privacy, but measuring and validating the actual privacy level of a machine learning pipeline or a trained machine learning model is not a trivial task. DP, for example, provides privacy guarantees in terms of ϵ and δ in the case of ϵ, δ)-DP and similar parameters for other DP regimes. While DP parameters give us the theoretical bounds of the DP of the final model, giving the end-users a more tangible indication of "how much" their data and/or model is private if they spent a certain amount of privacy budget is still a challenge. One can showcase the privacy level of DP and non-DP mechanisms by showing their resilience to different types of adversarial attacks, but finding the suitable forms of attack given the problem at hand, implementing and performing them against trained models, and keeping our evaluation up-to-date has its own set of challenges.

6.2.3 Combining Privacy and Security in Collaborative Learning. Privacy and security take a different approach to protecting information. We see that both have their uses, and while we see papers combining both in distributed learning, we unfortunately see few papers in CBL that combine cryptographic methods with privacy. Combining both security and privacy can be seen as creating a "double line of defense" that provided protection against overlapping but different threats. The two types of protection would be able to complement and strengthen each other. While we see a few papers that combine security and privacy for CBL [26], the majority do not. This is quite different from FL, where a majority of papers consider DP. It is important to consider that, due to the nature of CBL sharing data, vulnerabilities can be different. However, it is still vulnerable to attacks based on model outputs, such as reconstruction attacks, that DP offers effective protection against. Naturally, there is a payoff in accuracy and efficiency that needs to be considered, but the exact payoff in CBL should be more extensively studied.

6.2.4 Integrity of Data. While most research regarding protective measures of DDL has focused on providing data (and model) confidentiality, other issues distributed learning can bring forth should not be ignored, an important one being the integrity of data that the cooperating parties provide. We have covered several papers that combine confidentiality with integrity through verifiability [37, 92, 144], but we believe that it is essential when untrusted parties create a distributed model to have a form of verification of their data. Especially methods that include training could be sensitive to a poisoning attack, which involves the manipulation of training data to inject false information into the model, which can hinder the training process. Incorporating mechanisms for verification and accountability would be a critical step toward enhancing the security of collaborative learning systems. If we cannot guarantee the reliability of the data, then we can also not guarantee the reliability of the outcomes of the model. While we see some methods in CBL to include integrity, we believe this should be more widespread among all distributed learning methodologies.

6.2.5 Vertically Partitioned Data. We have encountered only a few papers in this survey that are able to perform distributed learning with vertically partitioned data. While vertically partitioned data is not as common of a data distribution as horizontally partitioned data, it can be common within in fields of medicine and finance. As these situations often deal with highly sensitive and possibly easily identifiable data, there must be proper security and privacy mechanisms in place. Most notably, within split learning, there has been some research into vertical split learning [20]. Unfortunately, many of these methods are shown to be vulnerable to attacks. More research into these vertically distributed data frameworks—also outside of split learning, as well as their protective measures—is recommended.

6.3 Emerging Areas of PSDDL

This section of the discussion focuses on privacy and security methods and frameworks, which were outside of the scope of Sections 4 and 5 of this survey, but we believe deserve attention, and they involve distributed frameworks, new security methods, and other aspects of distributed learning that will promote new areas of research in PSDDL.

6.3.1 Divergent Distributed Frameworks. Although DDL and CBL have been the dominant paradigm in distributed machine learning and despite the fact that they contain a wide variety of methods, there exist other distributed privacy-preserving deep learning methods in the literature that can not be classified under these umbrella terms. In this section, we review the prominent methods in this area. Cheng et al. [23] propose a decentralized deep network training scheme by proposing a differentially private iteration of the **Leader-Follower Elastic Averaging Stochastic Gradient Descent (LEASGD)** algorithm [22]. They apply calibrated Gaussian noise to the update rule of LEASGD, making each step of the algorithm (ϵ, δ)-differentially private. Yan et al. [148] designed a top-down hierarchical approach for deep learning on edge devices. The model is trained in a differentially private fashion using a Gaussian mechanism. Then, to tailor the trained model for usage on resource-constrained edge devices, the authors weight-prune the network and retrain it, again with DP using a Gaussian mechanism with a separate privacy budget, and publish the compressive models to the edge devices. In this setting, the data is centralized, but we have a distributed setting in which using DP helps preserve the privacy of the data on Learning Parties. Yuan et al. [152] propose a distributed learning method, combining local DP and SMC to train a discriminative model. They adopt the definition of label DP, i.e., only label information is considered sensitive. They propose two mechanisms, one based on the randomized response and one adapting the standard DP-SGD algorithm for label-DP.

6.3.2 Emerging Security Methods. While we mainly focused on the cryptographic methods of HE and SMC, more methods can achieve security. Two prominent examples of new methods are game theory [33] and blockchain [103]. Domingo-Ferrer et al. [33] set up a reputation management system with parties participating in FL with co-utility. This gives the parties an incentive to stick with the protocol. In Reference [103] they implement a verifiable and auditable FL framework based on the blockchain system. While not necessarily a new method, one of the main topics is quantum security. With the rise of quantum computers, many "standard" forms of encryption could be broken. While there has been a rise in research of quantum secure encryption, as of yet there has been little applied to distributed deep learning. Zuo et al. [163] focus on adapting secure aggregation to withstand quantum attacks.

6.3.3 Model Interpretations of DDL. Next to model training and inference, there are other aspects of deep learning that could be applied in a distributed manner, such as feature selection, explainability, and interpretability. Examples of these techniques in a distributed framework can be found in References [77, 111]. These methods would aid to create a more complete distributed learning setup. However, if these methods were utilized in DDL, then there could be more risk of data and model leakage. Therefore, it would be essential to adapt them to be private and secure.

7 Conclusion

In this work, we provided a high-level overview of secure and private distributed deep learning. We categorized different methods in this research area based on their distribution methodologies, namely, collaborative learning and distributed learning, provided a comparative summary of the different workflows, and highlighted promising methods and current research gaps. We provided background knowledge in privacy and security for this work to be self-contained while aiming toward a more general audience.

A Appendix

A.1 Protocol of Literature Evaluation

Our main sources of literature are peer-reviewed publications, including conference and journal papers that fall into our selection criteria. There exists research on preprint services such as arXiv,[1] which, although inside the scope of this survey, have not been through the peer review process. We have decided to mostly exclude these works unless warranted by substantial attention and number of citations.The initial queries for the selection process are: ***(("differential privacy" or "differentially private" or "privacy preserving") AND ("deep learning" OR "deep networks") and ("federated" OR "distributed" OR "decentralized")) OR (("secure" or "security" or "homomorphic encryption" or "multiparty computation") AND ("deep learning" OR "neural networks") and ("federated" OR "distributed"))***.

Due to the usually sophisticated nature of the deep learning methods and the additional complications arising from the addition of privacy-preserving measures to them, there are many details

[1]https://arxiv.org

to be considered for a fair and thorough comparison. We found the best approach would be to pay attention to both high-level as well as specific low-level details of the reviewed literature. Our initial set of review and comparison aspects include the prerequisite conditions for each method to work successfully, the distributed deep learning mechanism, the privacy and security mechanisms, and their use cases. Moreover, we only considered papers that had a relevant contribution to the privacy or security measures. We also aimed to only include papers that implemented and evaluated their setup experimentally, unless their importance to the field warranted otherwise. In our initial set of papers, we paid close attention to the aforementioned aspects of the collected literature and excluded the papers outside the scope of this research.

A.2 Basics of Differential Privacy

Differential Privacy to provide measures of privacy for algorithms by adding random perturbations to the data used and produced by the algorithms in different stages. It stops the adversarial party from inferring with a high level of confidence information about the training data.

A.2.1 Definition of Differential Privacy. Consider datasets S and S'. We define the distance of S and S', $d(S, S')$, as the minimum number of changes required for S to become S'. If $d(S, S') = 1$, then we call S) and S' adjacent or neighbor datasets. The random mechanism $M : D \rightarrow R$ with domain D and range R, which takes dataset s as an input and returns the random variable $M(S)$ will be (ϵ, δ)-*differentially private* for all neighbor pairs of $S, S' \in D$ with $d(S, S') = 1$ if for any subset of outputs $O \subseteq R$ there holds

$$P[M(S) \in O] \leq e^{\epsilon} P[M(S') \in O] + \delta, \tag{1}$$

in which ϵ is the privacy budget and δ is the probability of failure and controls the possibility of a ϵ-*differentially private* mechanism being broken with the probability of δ. Setting δ to zero would turn M into a strict ϵ-*differentially private* mechanism. The privacy budget ϵ controls the tradeoff between accuracy and privacy. Increasing ϵ will lead to weaker privacy and more leakage. A small ϵ will lead to better privacy but less accuracy of the mechanism M.

A.2.2 Sensitivity Function. Function $q(S)$ is defined as a *query* function that takes dataset S as an input and returns real values as an answer to the formulated query. The function q can return a number, a vector, or a categorical value.

Given query function q with non-categorical answers, $q(S) \in \mathbb{R}^n$, the *sensitivity* of q over adjacent datasets S and S' is defined as

$$s(q_{S,S'}, \|.\|) = max \, \|q(S) - q(S'\| \,, \tag{2}$$

in which $\|.\|$ is a norm function over q.

A.2.3 Composition and Domain of Differential Privacy Mechanisms. When designing differentially private machine learning models and algorithms, there are many cases in which we need to combine several differential privacy methods in our training process. One example would be deep neural networks, in which our aim is to apply the differential privacy mechanism M in different layers and thus, a sequential combination scheme is desired.

There are two theorems dealing with combination of differential privacy mechanism: sequential and parallel.

Parallel Theorem. If n random differential privacy mechanisms $M_i (1 \leq i \leq n)$ are applied to datasets $S_1, S_2, \ldots, S_n$ in a randomized parallel fashion, then the combination of mechanisms will be $(max(\epsilon_i, \epsilon_2, ...\epsilon_n), max(\delta_i, \delta_2, ...\delta_n))$-*differentially private.*

Sequential Theorem. If n random (ϵ, δ)-*differentially private* mechanisms $M_i(1 \leq i \leq n)$ are applied to datasets $S_1, S_2, \ldots, S_n$ in a sequential fashion, then the combination of mechanisms will be E, Δ-*differentially private* on the entire dataset, where $E = \sum_{i=1}^{n} \epsilon_i$ and $\Delta = \sum_{i=1}^{n} \delta_i$.

The Differential Privacy Domains. Randomized algorithms can achieve differential privacy in two domains: Global domain and Local domain. In **Global Differential Privacy (GDP)** scheme a global trusted curator receives non-perturbed data from involved parties in response to a query and applies random noise to the output of any queries being performed against the system. The randomized mechanism M is (ϵ, δ)-*differentially private* in global domain if it satisfies Equation (1). In **Local Differential Privacy (LDP)**, each node applies its own set of noise to the data and what the curator receives is the perturbed data [14].

A.3 Introduction—Differentially Private Deep Neural Networks (Non-distributed)

A.3.1 Differential Privacy Mechanisms.

Laplacian Differential Privacy. Laplacian differential privacy uses Laplacian perturbation to add noise to the data. The random function $\tilde{q}$ is defined as $\tilde{q}(S) = q(S) + \eta$ over query function q and is ϵ-*differentially private.* The random variable η in Laplacian differential privacy is defined with probability density function $p(\eta) \propto e^{\|\eta\|/s(q, \|.\|)}$.

Gaussian Differential Privacy. Gaussian differential privacy has a similar mechanism to Laplacian differential privacy, replacing Laplacian perturbation with Gaussian perturbation. In Gaussian differential privacy η is defined as a random variable from distribution $N(0, \frac{2}{\epsilon^2}(s(q, \|.\|))^2 log\frac{2}{\delta})$

Randomized Response-based Differential Privacy. Randomized Response, [141] is a method designed to fix the problem of evasive answer bias of binary surveys. This method makes use of two coins, returning "true" or "false." The coins are biased with the probability p of returning "true." If the first coin toss returns "true," then the answer is relayed unchanged; if, however, it returns "false," then "yes" is returned when the second coin toss yields "true" and "no" when it returns "false."

To achieve ϵ-*differential privacy* the probability p of the biased coin is defined as

$$p = \frac{e^{\epsilon}}{1 + e^{\epsilon}} \tag{3}$$

.[140]

Functional Mechanism for Differentially Private Regression. ***Functional Mechanism (FM)*** [155] is a framework for differentially private regression. Based on Laplacian differential privacy (see Section A.3.1), it aims at perturbing the optimization goal of the regression operation instead of perturbing the results of the regression analysis.

Considering function $q(w)$ over dataset D, input parameter w is considered to be a vector of the length d. Let $\phi(w) = w_1^{c_1}.w_2^{c_2}...w_d^{c_d}$ where $c_1, c_2, \ldots, c_d \in N$ and $\Phi_j = \left\{w_1^{c_1}.w_2^{c_2}...w_n^{c_d} \mid \sum_{l=1}^{d} c_l = j\right\}$ where $j \in N$. Based on the Stone-Weierstrass theorem [27], continuous and differentiable form of $q_D(w)$ can be written in polynomial form $\sum_{j=0}^{J} \sum_{\phi \in \Phi_j} \sum_{x_i \in D} \lambda_{\Phi_{x_i}} \Phi(w)$ in which $J \in [0, \infty]$ and values $\lambda_{\Phi_{x_i}} \Phi(w) \in R$ are the coefficients of the polynomial form of $\phi(w)$.

FM derives the ϵ-*differentially private* form of $q_D(w)$ by perturbing polynomial coefficients $\lambda(\phi)$ using Laplacian noise $Lap(\Delta/\epsilon)$ where $\Delta = 2max_x \sum_{j=0}^{J} \sum_{\phi \in \Phi_j} \left\|\lambda_{\Phi_x}\right\| 1$.

A.4 Homomorphic Encryption

Homomorphic encryption (HE) is a type of encryption protocol that allows for computation on the ciphertext without the necessity of decryption of the ciphertext to perform the calculation. The possible calculations that can be performed on the ciphertexts depend on the scheme. **Fully homomorphic encryption (FHE)** schemes allow for the homomorphic evaluation of all arbitrary functions. To achieve this, it is sufficient to allow only addition and multiplication operations, as they make up all possible finite calculations and an unlimited number of operations. Not all HE schemes allow for these calculations in unlimited numbers. When performing the calculations, noise is added to the encryption, until a point where the noise. Only with the development of bootstrapping, a technique to limit noise by re-encrypting the message by Gentry [46], FHE became possible, while still very computationally costly.

Before the development of bootstrapping, the schemes allowed either one type of operation or a limited number of operations on the encrypted data. The different schemes can therefore be categorized as follows: **Partially Homomorphic Encryption (PHE)**, which allows for one type of operation (addition or multiplication), but an unlimited number of times; **Somewhat Homomorphic Encryption (SWHE)** allows both types of operations, but only a limited number of times; **Fully Homomorphic Encryption (FHE)** allows both operations for an unlimited number of times. There are also subtypes of FHE, such as leveled FHE, where the depth of the operations is bound (for efficiency's sake) and multikey FHE, which allows for the different inputs to be encrypted with different keys.

An HE scheme generally consists of four operations: KeyGen, Enc, Dec, and Eval. KeyGen generates the key in case of symmetric encryption or the key pair in case of asymmetric encryption. KeyGen, Enc(ryption) and Dec(ryption) are comparable to their counterparts in non-homomorphic encryption schemes. Eval is the operation that is HE-specific. Eval performs a function over the ciphertexts without seeing the unencrypted messages. Eval has ciphertexts as input and outputs. The most essential point in this evaluation is that the format of the ciphertexts needs to be preserved to be decrypted correctly. Moreover, the size of the ciphertext should be constant to support FHE and thus an unlimited number of operations, as else it would require more resources, which would limit the number of operations. HE schemes can be designed for symmetric encryption, where the same keys are used for encryption and decryption, as well as asymmetric, where different keys are used.

A.4.1 Partially Homomorphic Encryption. A cryptosystem is considered partially homomorphic if it supports either evaluation of the ciphertexts by adding or multiplying of but not both operations at the same time. These schemes can be **additive Homomorphic Encryption (AHE)**, supporting addition, or **multiplicative Homomorphic Encryption (MHE)**, supporting multiplication. Some of the PHE cryptosystems:

— Paillier Scheme (AHE) [98]: an asymmetric algorithm for public key cryptography. It allows for additional HE and is based on the problem of computing nth residue classes, which is believed to be computationally difficult.
— RSA Scheme (MHE) [115] is the asymmetric cryptosystem widely used for communication. In the RSA system, the public key is created using two large two large prime numbers, with the prime numbers kept secret. Anyone can use this key to encrypt, but the messages can only be decrypted by someone who knows the prime numbers. RSA was shown to be AHE by Reference [98].
— ElGamal cryptosystem (MHE) [40] is also asymmetric key encryption algorithm for public key cryptography but is multiplicative HE. ElGamal is based on the DiffieHellman key exchange [32]

A.4.2 Somewhat Homomorphic Encryption. SWHE supports both multiplication and addition, but only a limited number of times. Before the development of bootstrapping by Gentry[46], an important step towards an FHE scheme was introduced by **Boneh-Goh-Nissim (BGN)** [14]. BGN supports an arbitrary number of additions and one multiplication by keeping the ciphertext size constant. After Gentry's bootstrapping allowed for FHE, most SWHE schemes can be considered the part of the FHE schemes rather than a separate scheme. As such, a leveled FHE could be considered the same as an SWHE. Therefore, we currently do not distinguish between SWHE and (L)FHE schemes.

A.4.3 Fully Homomorphic Encryption. Fully homomorphic encryption first became possible after the development of Gentry's bootstrapping [46]. The idea behind bootstrapping is to use a homomorphic decryption circuit to decrypt the ciphertext, perform an operation on the plaintext, and then re-encrypt the result. This way, the new ciphertext encrypts the result of the computation without leaking information about the plaintext. Without bootstrapping, the noise in the ciphertexts produced by FHE operations would accumulate over time, eventually rendering the encrypted data useless. So, in their essence, FHE schemes are SWHE schemes, where a squashing method to reduce the circuit depth of the decryption algorithm is introduced, and bootstrapping is applied to obtain fresh ciphertext to avoid getting too much noise.

While FHE protocols have been developed, they are computationally expensive, especially as the complexity of the computation function increases. This has led to the development of **leveled FHE (LFHE)**, where the complexity of the computations are bounded by depth. For example, a polynomial would be not possible. Most FHE systems used in current secure distributed learning frameworks are actually leveled FHE. Most of these methods are based on learning with errors problem. A short overview of these methods is as follows:

— **Brakerski-Gentry-Vaikuntanathan (BGV)** [18] is one of the first practical FHE schemes, introduced in 2011, and is designed for computations on integers.
— **Yet Another Somewhat Homomorphic Encryption (YASHE)** [55] scheme was a fully homomorphic encryption scheme that is based on the **learning with errors (LWE)** problem. It has been shown to have vulnerabilities [4, 71]
— **Brakerski-Fan-Vercauteren (BFV)** scheme [17] is a relatively efficient FHE scheme. It supports both integer and polynomial plaintexts.
— **Cheon-Kim-Kim-Song (CKKS)** scheme [24] was designed for FHE over real or complex numbers and is considered to have high computational efficiency. It is based on the ring-LWE problem.
— **Fan-Vercauteren (FV)** scheme[43] is another RLWE-based FHE scheme but is known for its simplicity and its ability to support both integer and binary plaintexts.
— **Gentry-Sahai-Waters (GSW)** scheme [48] is a scheme that focuses on improving bootstrapping speed instead of limiting noise in operations. It only supports binary plaintexts, which can limit its usefulness in certain applications. **FHE over the Torus (TFHE)** [25] is based on this scheme.

While many different FHE schemes have been developed, many have not been publicly available for implementation. Two of the often used publicly available implementations are HElib [56] and SEAL [121] Helib runs on BGV, and SEAL used to run on YASHE but now implements FV.

A.4.4 Multikey FHE. Multikey FHE is designed to provide evaluation on ciphertexts that are encrypted with different keys; this means that each participant can encrypt data with their own public key and a homomorphic evaluation can be performed on these ciphertexts. The only interaction required between participants is to create a "joint secret key," which is obtained by using all involved secret keys. This joint secret key is then used to decrypt the homomorphically evaluated ciphertext is decrypted by using. Several FHE schemes have been adapted for multikey-FHE, inlcuding CKKS [21], GSW[47], among others.

A.5 Secure Multiparty Encryption

Secure multiparty computation (SPMC) methods make a distinction between **multiparty computation (MPC)** and **two party computation (2PC)**. While other protocols have been developed, the most common methods are Garbled circuits, based on oblivious transfer, for 2PC and secret sharing for MPC.

A.5.1 Oblivious Transfer. A main building block for various types of encryption is Oblivious Transfer, first developed by Rabin in 1981 [108]. The goal of the OT encryption protocol is secure sending of information between a sender (S) and a receiver (R). The standard OT protocol is the 1-out-of-n **oblivious transfer (OT)**, where (R) gets one element out of n but without (S) knowing which element. The oblivious transfer protocol is usually executed through an assymetric cryptosystem like the RSA system [115]. This can be extended to a (k-out-of-n protocol), where the receiver obtains a set of k messages from the message set n, where $1 \leqq k \leqq n$.

Dot product triplets, also known as inner product triplets, are a way of securely computing the dot product of two vectors without revealing the vectors themselves. Given two vectors x and y, a dot product triplet consists of three values: (a, b, c), where a and b are random vectors with the same length as x and y, and c is the dot product of a and b. The triplet is constructed in such a way that each party holds a share of a, b, and c, and they can collaboratively compute the dot product of x and y without revealing x or y to each other. The computation involves each party multiplying their share of a and b and then summing the resulting values, which gives them a share of the dot product of x and y. Dot triplet products are similar to the multiplication products of OT, providing an opportunity for the use of OT in secure DDL.

A.5.2 Garbled Circuits. Based on OT, Yao's **garbled circuits (GC)** was the first technique that enables secure two-party computation [149]. A garbled circuit is a circuit (a collection of logical gates that compute a function), where the inputs and outputs of the circuit are encrypted. The inputs are encrypted in such a way that each party can evaluate the circuit on their inputs while not learning anything about the other parties' inputs or the any intermediate values. Garbled circuits are designed so the parties only learn the final output of the circuit. Garbled circuits work by taking a computation and expressing it as a set of logic gates and securely executing the operations (AND, OR, NOT) in the circuit. In distributed deep learning, Yao's GC protocol, where the two parties can jointly calculate the outcome of a function, is usually implemented. GC works with two mistrusting parties is secure against a semi-honest adversary.

A protocol between Alice and Bob about function *f(x,y*i without revealing their respective inputs x and y is formalized as follows:

(1) Both parties agree on a way to express f as a Boolean circuit. Alice garbles the circuit f to $\hat{\mathrm{f}}$. She sends the GC $\hat{\mathrm{f}}$ to Bob as well as her own "garbled input" $\hat{\mathrm{x}}$.
(2) Alice at this point knows how to encode any input for f into a "garbled" input, while only Bob knows his private input y. So, using oblivious transfer, Bob is able to obtain a garbled version $\hat{\mathrm{y}}$ without Alice learning what y was.
(3) Bob has now the garbled circuit $\hat{\mathrm{f}}$, both garbled input $\hat{\mathrm{x}}$,$\hat{\mathrm{y}}$ for that circuit. He runs the evaluation and learns $f(x, y)$ and reveals this to Alice.

A.5.3 Secret Sharing. To perform secure computation in a situation with more than two participants, **secret sharing (SS)** is often implemented. Here, one party has a secret that it distributes among a number of parties so none of the parties alone can recover the secret. This way, there is a need for cooperation between parties to discover the secret. Secret sharing is often the basis of secure **multiparty computation (MPC)** and can be homomorphic as well.

Additive secret sharing Here, one party has a secret that it distributes among a number of parties so none of the parties alone can recover the secret. This way, there is a need for cooperation between parties to discover the secret. Secret sharing works as follows considering an $A(t, n)$ secret sharing scheme: S would be a sharing function that takes the secret s as input and creates n secret shares: $S(s) = (s1, ..., sn)$. A minimum of the n parties are needed for the protocol. It is required that S cannot be recovered from a set of $s - 1$ secret shares.

Shamir's secret sharing (SSS) relies on the idea of using a polynomial to convey the secret, first introduced by Shamir [123]. In this scheme, any t out of n shares may be used to recover the secret. The idea is that if you have a set of t points, then you can fit a unique polynomial of degree (t-1) that these points will lie in. As such, two points can define a straight line, three points can define a quadratic function, four points a cubic curve, and so on. With SSS, a function would be made where the secret is the first coefficient of the polynomial, with the remaining coefficients picked at random. Then, n points are given to each of the participants. When at least t participants share their points, the function, and thereby the secret, can be recovered. In some cases, FTT [19] is used as the method to find this polynomial.

A.5.4 Functional Encryption. Functional encryption is a relatively new area of encryption that goes against the current idea of standard encryption. For functional encryption systems, the decryption key does not allow the user to learn the encrypted data, but instead a function of the encrypted data. Functional encryption allows authorized users to derive specific decryption keys that only reveal a specific function of the encrypted data, instead of revealing the entire decrypted data. As such, it allows different users to have access to different functions of the same encrypted data according to their specific access privileges. This method can allow for more fine-grained access control and privacy-preserving data sharing

References

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

[2] Nitin Agrawal, Ali Shahin Shamsabadi, Matt J. Kusner, and Adrià Gascón. 2019. QUOTIENT: Two-party secure neural network training and prediction. *CoRR* abs/1907.03372 (2019).

[3] Ahmad Al Badawi, Chao Jin, Jie Lin, Chan Fook Mun, Sim Jun Jie, Benjamin Hong Meng Tan, Xiao Nan, Khin Mi Mi Aung, and Vijay Ramaseshan Chandrasekhar. 2021. Towards the AlexNet moment for homomorphic encryption: HCNN, the first homomorphic CNN on encrypted data with GPUs. *IEEE Trans. Emerg. Topics Comput.* 9, 3 (2021), 1330–1343. DOI: https://doi.org/10.1109/TETC.2020.3014636

[4] Martin Albrecht, Shi Bai, and Léo Ducas. 2016. A subfield lattice attack on overstretched NTRU assumptions: Cryptanalysis of some FHE and graded encoding schemes. Cryptology ePrint Archive, Paper 2016/127. Retrieved from https://eprint.iacr.org/2016/127

[5] Emmanuel Antwi-Boasiako, Shijie Zhou, Yongjian Liao, Qihe Liu, Yuyu Wang, and Kwabena Owusu-Agyemang. 2021. Privacy preservation in distributed deep learning: A survey on distributed deep learning, privacy preservation techniques used and interesting research directions. *J. Inf. Secur. Applic.* 61 (2021), 102949. DOI: https://doi.org/10.1016/j.jisa.2021.102949

[6] Gilad Asharov, Yehuda Lindell, Thomas Schneider, and Michael Zohner. 2013. More efficient oblivious transfer and extensions for faster secure computation. Cryptology ePrint Archive, Paper 2013/552. DOI: https://doi.org/10.1145/2508859.2516738

[7] Louis J. M. Aslett, Pedro M. Esperança, and Chris C. Holmes. 2015. Encrypted statistical machine learning: new privacy preserving methods. DOI: https://doi.org/10.48550/ARXIV.1508.06845

[8] Borja Balle, James Bell, Adrià Gascón, and Kobbi Nissim. 2019. The privacy blanket of the shuffle model. In *Advances in Cryptology–CRYPTO 2019: 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18–22, 2019, Proceedings, Part II 39*. Springer, 638–667.

[9] Brett K. Beaulieu-Jones, William Yuan, Samuel G. Finlayson, and Zhiwei Steven Wu. 2018. Privacy-preserving distributed deep learning for clinical data. *arXiv preprint arXiv:1812.01484* (2018).

[10] James Bell, K. A. Bonawitz, Adrià Gascón, Tancrède Lepoint, and Mariana Raykova. 2020. Secure single-server aggregation with (poly)logarithmic overhead. Cryptology ePrint Archive, Paper 2020/704. DOI: https://doi.org/10.1145/3372297.3417885

[11] Fabian Boemer, Anamaria Costache, Rosario Cammarota, and Casimir Wierzynski. 2019. nGraph-HE2: A high-throughput framework for neural network inference on encrypted data. *CoRR* abs/1908.04172 (2019).

[12] Fabian Boemer, Yixing Lao, Rosario Cammarota, and Casimir Wierzynski. 2019. NGraph-HE: A graph compiler for deep learning on homomorphically encrypted data. In *16th ACM International Conference on Computing Frontiers (CF'19)*. Association for Computing Machinery, New York, NY, USA, 3–13. DOI: https://doi.org/10.1145/3310273.3323047

[13] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. 2017. Practical secure aggregation for privacy-preserving machine learning. In *ACM SIGSAC Conference on Computer and Communications Security (CCS'17)*. Association for Computing Machinery, New York, NY, USA, 1175–1191. DOI: https://doi.org/10.1145/3133956.3133982

[14] D. Boneh, E. Goh, and Kobbi Nissim. 2005. Evaluating 2- DNF formulas on ciphertexts." In *2nd Conference on Theory of Cryptography*. 325–342.

[15] Florian Bourse, Michele Minelli, Matthias Minihold, and Pascal Paillier. 2018. Fast homomorphic evaluation of deep discretized neural networks. In *Advances in Cryptology – CRYPTO 2018*, Hovav Shacham and Alexandra Boldyreva (Eds.). Springer International Publishing, Cham, 483–512.

[16] Elette Boyle, Niv Gilboa, and Yuval Ishai. 2015. Function secret sharing. In *Advances in Cryptology-EUROCRYPT 2015: 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part II*. Springer, 337–367.

[17] Zvika Brakerski, Jintai Fan, and Frederik Vercauteren. 2012. Fully homomorphic encryption without bootstrapping. In *3rd Innovations in Theoretical Computer Science Conference*. ACM, 309–325.

[18] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. 2014. (Leveled) fully homomorphic encryption without bootstrapping. *ACM Trans. Comput. Theor.* 6, 3, Article 13 (July 2014), 36 pages. DOI: https://doi.org/10.1145/2633600

[19] E. O. Brigham and R. E. Morrow. 1967. The fast Fourier transform. *IEEE Spect.* 4, 12 (1967), 63–70. DOI: https://doi.org/10.1109/MSPEC.1967.5217220

[20] Iker Ceballos, Vivek Sharma, Eduardo Mugica, Abhishek Singh, Alberto Roman, Praneeth Vepakomma, and Ramesh Raskar. 2020. SplitNN-driven vertical partitioning. *arXiv preprint arXiv:2008.04137* (2020).

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

[22] Hsin-Pai Cheng, Patrick Yu, Haojing Hu, Feng Yan, Shiyu Li, Hai Li, and Yiran Chen. 2018. LEASGD: An efficient and privacy-preserving decentralized algorithm for distributed learning. *arXiv preprint arXiv:1811.11124* (2018).

[23] Hsin-Pai Cheng, Patrick Yu, Haojing Hu, Syed Zawad, Feng Yan, Shiyu Li, Hai Li, and Yiran Chen. 2019. Towards decentralized deep learning with differential privacy. In *International Conference on Cloud Computing*. Springer, 130–145.

[24] Jung Hee Cheon, Minjia Kim, Hyunsoo Kim, and Yongsoo Song. 2018. Homomorphic encryption for arithmetic of approximate numbers. In *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 523–552.

[25] N. Gama, M. Georgieva, M. Izabachène, and I. Chillotti. 2016. TFHE: Fast fully homomorphic encryption over the torus. In *IEEE Symposium on Security and Privacy*.

[26] Edward Chou, Josh Beal, Daniel Levy, Serena Yeung, Albert Haque, and Li Fei-Fei. 2018. Faster CryptoNets: Leveraging sparsity for real-world encrypted inference. DOI: https://doi.org/10.48550/ARXIV.1811.09953

[27] Neil E. Cotter. 1990. The Stone-Weierstrass theorem and its application to neural networks. *IEEE Trans. Neural Netw.* 1, 4 (1990), 290–295.

[28] Anders Dalskov, Daniel E. Escudero, and Marcel Keller. 2020. Fantastic four: Honest-majority four-party secure computation with malicious security. *IACR Cryptol. ePrint Arch.* 2020 (2020), 1330.

[29] D. Demmler, T. Schneider, and M. Zohner. 2015. ABY—A framework for efficient mixed-protocol secure two-party computation. Internet Society. DOI: https://doi.org/10.14722/ndss.2015.23113

[30] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. ImageNet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'09)*. IEEE, 248–255.

[31] Li Deng. 2012. The MNIST database of handwritten digit images for machine learning research [best of the web]. *IEEE Signal Process. Mag.* 29, 6 (2012), 141–142.

[32] Whitfield Diffie and Martin Hellman. 1976. New directions in cryptography. *IEEE Trans. Inf. Theor.* 22, 6 (1976), 644–654.

[33] Josep Domingo-Ferrer, Alberto Blanco-Justicia, Jesús Manjón, and David Sánchez. 2021. Secure and privacy-preserving federated learning via co-utility. DOI: https://doi.org/10.48550/ARXIV.2108.01913

[34] Jinshuo Dong, Aaron Roth, and Weijie J. Su. 2019. Gaussian differential privacy. *arXiv preprint arXiv:1905.02383* (2019).

[35] Nathan Dowlin, Ran Gilad-Bachrach, Kim Laine, Kristin Lauter, Michael Naehrig, John Wernsing, and Microsoft Research. 2016. *CryptoNets: Applying Neural Networks to Encrypted Data with High Throughput and Accuracy.* Technical Report. Retrieved from http://sealcrypto.codeplex.com

[36] Jia Duan, Jiantao Zhou, and Yuanman Li. 2020. Privacy-preserving distributed deep learning based on secret sharing. *Inf. Sci.* 527 (2020), 108–127. DOI: https://doi.org/10.1016/j.ins.2020.03.074

[37] Jia Duan, Jiantao Zhou, Li Yuanman, and Caishi Huang. 2022. Privacy-preserving and verifiable deep learning inference based on secret sharing. *Neurocomputing* 483 (01 2022). DOI: https://doi.org/10.1016/j.neucom.2022.01.061

[38] John C. Duchi, Michael I. Jordan, and Martin J. Wainwright. 2018. Minimax optimal procedures for locally private estimation. *J. Am. Stat. Assoc.* 113, 521 (2018), 182–201.

[39] Cynthia Dwork, Aaron Roth, et al. 2014. The algorithmic foundations of differential privacy. *Found. Trends Theoret. Comput. Sci.* 9, 3-4 (2014), 211–407.

[40] Taher Elgamal. 1985. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Advances in Cryptology Conference (CRYPTO'84).* Springer, 10–18.

[41] David Enthoven and Zaid Al-Ars. 2021. *An Overview of Federated Deep Learning Privacy Attacks and Defensive Strategies.* Springer International Publishing, Cham, 173–196. DOI: https://doi.org/10.1007/978-3-030-70604-3_8

[42] Ege Erdoğan, Alptekin Küpçü, and A. Ercüment Çiçek. 2022. Unsplit: Data-oblivious model inversion, model stealing, and label inference attacks against split learning. In *21st Workshop on Privacy in the Electronic Society.* 115–124.

[43] Junfeng Fan and Frederik Vercauteren. 2012. Somewhat practical fully homomorphic encryption. In *International Workshop on Public Key Cryptography.* Springer, 17–34.

[44] General Data Protection Regulation (GDPR). 2018. General data protection regulation (GDPR)—Final text neatly arranged.

[45] Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. 2020. Inverting gradients—How easy is it to break privacy in federated learning? *Advan. Neural Inf. Process. Syst.* 33 (2020), 16937–16947.

[46] Craig Gentry. 2009. A fully homomorphic encryption scheme.

[47] Craig Gentry, Shai Halevi, Nigel P. Smart, and Shuhong Wang. 2017. Multi-key homomorphic encryption from learning with errors. In *Annual Cryptology Conference.* Springer, 345–373.

[48] Craig Gentry, Amit Sahai, and Brent Waters. 2013. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *Annual Cryptology Conference.* Springer, 75–92.

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

[50] Zahra Ghodsi, Tianyu Gu, and Siddharth Garg. 2017. SafetyNets: Verifiable execution of deep neural networks on an untrusted cloud. *CoRR* abs/1706.10268 (2017).

[51] Parham Gohari, Bo Chen, Bo Wu, Matthew Hale, and Ufuk Topcu. 2021. Privacy-preserving teacher-student deep reinforcement learning. arXiv:2102.09599 (2021).

[52] Oded Goldreich, Silvio Micali, and Avi Wigderson. 2019. How to play any mental game, or a completeness theorem for protocols with honest majority. In *Providing Sound Foundations for Cryptography.*

[53] Thore Graepel, Kristin Lauter, and Michael Naehrig. 2012. ML confidential: Machine learning on encrypted data. Cryptology ePrint Archive, Paper 2012/323. Retrieved from https://eprint.iacr.org/2012/323

[54] Trung Ha, Tran Khanh Dang, Tran Tri Dang, Tuan Anh Truong, and Manh Tuan Nguyen. 2019. Differential privacy in deep learning: An overview. In *International Conference on Advanced Computing and Applications (ACOMP'19).* IEEE, 97–102.

[55] Shai Halevi and Victor Shoup. 2013. An algorithm for fully homomorphic encryption over the torus. In *Annual Cryptology Conference.* Springer, 505–524.

[56] Shai Halevi and Victor Shoup. 2014. Algorithms in HElib. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques.* Springer, 554–571.

[57] Gang Han, Tiantian Zhang, Yinghui Zhang, Guowen Xu, Jianfei Sun, and Jin Cao. 2022. Verifiable and privacy preserving federated learning without fully trusted centers. *J. Amb. Intell. Human. Comput.* 13 (03 2022), 1–11. DOI: https://doi.org/10.1007/s12652-020-02664-x

[58] Meng Hao, Hongwei Li, Guowen Xu, Sen Liu, and Haomiao Yang. 2019. Towards efficient and privacy-preserving federated deep learning. In *IEEE International Conference on Communications (ICC'19).* IEEE, 1–6.

[59] Valentin Hartmann and Robert West. 2019. Privacy-preserving distributed learning with secret gradient descent. *arXiv preprint arXiv:1906.11993* (2019).

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

[61] Ehsan Hesamifard, Daniel Takabi, Mehdi Ghasemi, and Rebecca Wright. 2018. Privacy-preserving machine learning as a service. *Proc. Privac. Enhanc. Technol.* 2018 (06 2018), 123–142. DOI: https://doi.org/10.1515/popets-2018-0024

[62] Ehsan Hesamifard, Hassan Takabi, and Mehdi Ghasemi. 2017. CryptoDL: Deep neural networks over encrypted data. *CoRR* abs/1711.05189 (2017).

[63] Hongsheng Hu, Zoran Salcic, Lichao Sun, Gillian Dobbie, Philip S. Yu, and Xuyun Zhang. 2022. Membership inference attacks on machine learning: A survey. *ACM Comput. Surv.* 54, 11s (2022), 1–37.

[64] Zonghao Huang, Rui Hu, Yuanxiong Guo, Eric Chan-Tin, and Yanmin Gong. 2020. DP-ADMM: ADMM-based distributed learning with differential privacy. *IEEE Trans. Inf. Forens. Secur.* 15 (2020), 1002–1012. DOI: https://doi.org/10.1109/TIFS.2019.2931068

[65] Alberto Ibarrondo and Melek Önen. 2018. FHE-compatible batch normalization for privacy preserving deep learning. In *Data Privacy Management, Cryptocurrencies and Blockchain Technology*, Joaquin Garcia-Alfaro, Jordi Herrera-Joancomartí, Giovanni Livraga, and Ruben Rios (Eds.). Springer International Publishing, Cham, 389–404.

[66] C. Juvekar, V. Vaikuntanathan, and A. Chandrakasan. 2018. Gazelle: A low latency framework for secure neural network inference. In *IACR Cryptol. ePrint Arch.*

[67] Swanand Kadhe, Nived Rajaraman, Onur Ozan Koyluoglu, and Kannan Ramchandran. 2020. FastSecAgg: Scalable secure aggregation for privacy-preserving federated learning. *CoRR* abs/2009.11248 (2020).

[68] Peter Kairouz, Brendan McMahan, Shuang Song, Om Thakkar, Abhradeep Thakurta, and Zheng Xu. 2021. *Practical and Private (Deep) Learning without Sampling or Shuffling*. Technical Report. arXiv:2103.00039v1

[69] Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. 2021. Advances and open problems in federated learning. *Found. Trends Mach. Learn.* 14, 1–2 (2021), 1–210.

[70] Muah Kim, Onur Günlü, and Rafael F. Schaefer. 2021. *Federated Learning with Local Differential Privacy: Trade-offs between Privacy, Utility, and Communication*. Technical Report. arXiv:2102.04737v1

[71] Paul Kirchner and Pierre-Alain Fouque. 2017. Revisiting lattice attacks on overstretched NTRU parameters. In *Advances in Cryptology – EUROCRYPT 2017*, Jean-Sébastien Coron and Jesper Buus Nielsen (Eds.). Springer International Publishing, Cham, 3–26.

[72] Nishat Koti, Mahak Pancholi, Arpita Patra, and Ajith Suresh. 2020. SWIFT: Super-fast and robust privacy-preserving machine learning. *CoRR* abs/2005.10296 (2020).

[73] Nishat Koti, Arpita Patra, Rahul Rachuri, and Ajith Suresh. 2021. Tetrad: Actively secure 4PC for secure training and inference. *CoRR* abs/2106.02850 (2021).

[74] Alex Krizhevsky, Geoffrey Hinton, et al. 2009. Learning multiple layers of features from tiny images. (2009).

[75] Owusu Agyemang Kwabena, Zhiguang Zhen Qin, Zhiguang Zhen Qin, and Tianming Zhuang. 2019. MSCryptoNet multi-scheme privacy-preserving deep learning in cloud computing. 29344–29354. DOI: https://doi.org/10.1109/ACCESS.2019.2901219

[76] Joon-Woo Lee, Hyungchul Kang, Yongwoo Lee, Woosuk Choi, Jieun Eom, Maxim Deryabin, Eunsang Lee, Junghyun Lee, Donghoon Yoo, Young-Sik Kim, and Jong-Seon No. 2022. Privacy-preserving machine learning with fully homomorphic encryption for deep neural network. *IEEE Access* 10 (2022), 30039–30054. DOI: https://doi.org/10.1109/ACCESS.2022.3159694

[77] Anran Li, Rui Liu, Ming Hu, Luu Anh Tuan, and Han Yu. 2023. Towards interpretable federated learning. *arXiv preprint arXiv:2302.13473* (2023).

[78] Oscar Li, Jiankai Sun, Xin Yang, Weihao Gao, Hongyi Zhang, Junyuan Xie, Virginia Smith, and Chong Wang. 2021. Label leakage and protection in two-party split learning. *arXiv preprint arXiv:2102.08504* (2021).

[79] Richard Lindner and Chris Peikert. 2011. Better key sizes (and attacks) for LWE-based encryption. In *Topics in Cryptology–CT-RSA 2011: The Cryptographers' Track at the RSA Conference 2011, San Francisco, CA, USA, February 14-18, 2011. Proceedings*. Springer, 319–339.

[80] Bo Liu, Ming Ding, Sina Shaham, Wenny Rahayu, Farhad Farokhi, and Zihuai Lin. 2021. When machine learning meets privacy: A survey and outlook. *ACM Comput. Surv.* 54, 2 (2021), 1–36.

[81] Jian Liu, Mika Juuti, Yao Lu, and N. Asokan. 2017. Oblivious neural network predictions via MiniONN transformations. In *ACM SIGSAC Conference on Computer and Communications Security (CCS'17)*. Association for Computing Machinery, New York, NY, USA, 619–631. DOI: https://doi.org/10.1145/3133956.3134056

[82] Xiaoyuan Liu, Hongwei Li, Guowen Xu, Rongxing Lu, and Miao He. 2020. Adaptive privacy-preserving federated learning. *Peer-to-peer Netw. Applic.* 13, 6 (2020), 2356–2366.

[83] Xiaoyuan Liu, Hongwei Li, Guowen Xu, Rongxing Lu, and Miao He. 2020. Adaptive privacy-preserving federated learning. *Peer-to-peer Netw. Applic.* 13, 6 (Nov. 2020), 2356–2366. DOI: https://doi.org/10.1007/s12083-019-00869-2

[84] Yang Liu, Tao Fan, Tianjian Chen, Qian Xu, and Qiang Yang. 2021. Fate: An industrial grade platform for collaborative learning with data protection. *J. Mach. Learn. Res.* 22, 1 (2021), 10320–10325.

[85] Guillermo Lloret-Talavera, Marc Jorda, Harald Servat, Fabian Boemer, Chetan Chauhan, Shigeki Tomishima, Nilesh N. Shah, and Antonio J. Pena. 2022. Enabling homomorphically encrypted inference for large DNN models. *IEEE Trans. Comput.* 71, 5 (2022), 1145–1155. DOI: https://doi.org/10.1109/TC.2021.3076123

[86] Yunhui Long, Suxin Lin, Zhuolin Yang, Carl A. Gunter, Han Liu, and Bo Li. 2019. Scalable differentially private data generation via private aggregation of teacher ensembles. (2019).

[87] Lingjuan Lyu, Han Yu, Xingjun Ma, Chen Chen, Lichao Sun, Jun Zhao, Qiang Yang, and Philip S. Yu. 2022. Privacy and robustness in federated learning: Attacks and defenses. *IEEE Trans. Neural Netw. Learn. Syst.* (2022).

[88] Jing Ma, Si-Ahmed Naas, Stephan Sigg, and Xixiang Lyu. 2022. Privacy-preserving federated learning based on multi-key homomorphic encryption. *Int. J. Intell. Syst.* 37, 9 (2022), 5880–5901. DOI: https://doi.org/10.1002/int.22818

[89] Mohammad Malekzadeh, Burak Hasircioglu, Nitish Mital, Kunal Katarya, Emre Ozfatura, and Deniz Gündüz. 2021. *Dopamine: Differentially Private Federated Learning on Medical Data.* Technical Report. 21 pages. Retrieved from https://www.kaggle.com/c/aptos2019-blindness-detection/notebooks

[90] Lourdes Martínez-Villaseñor, Hiram Ponce, Jorge Brieva, Ernesto Moya-Albor, José Núñez-Martínez, and Carlos Peñafort-Asturiano. 2019. UP-fall detection dataset: A multimodal approach. *Sensors* 19, 9 (2019), 1988.

[91] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics.* PMLR, 1273–1282.

[92] Pratyush Mishra, Ryan Lehmkuhl, Akshayaram Srinivasan, Wenting Zheng, and Raluca Ada Popa. 2020. Delphi: A cryptographic inference service for neural networks. Cryptology ePrint Archive, Paper 2020/050. Retrieved from https://eprint.iacr.org/2020/050

[93] Payman Mohassel and Peter Rindal. 2018. ABY3: A mixed protocol framework for machine learning. Cryptology ePrint Archive, Paper 2018/403. Retrieved from https://eprint.iacr.org/2018/403

[94] Payman Mohassel and Yupeng Zhang. 2017. SecureML: A system for scalable privacy-preserving machine learning. In *IEEE Symposium on Security and Privacy.* Institute of Electrical and Electronics Engineers Inc., 19–38. DOI: https://doi.org/10.1109/SP.2017.12

[95] Seung Ho Na, Hyeong Gwon Hong, Junmo Kim, and Seungwon Shin. 2022. Closing the loophole: Rethinking reconstruction attacks in federated learning from a privacy standpoint. In *38th Annual Computer Security Applications Conference.* 332–345.

[96] Karthik Nandakumar, Nalini Ratha, Sharath Pankanti, and Shai Halevi. 2019. Towards deep neural network training on encrypted data. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW'19).* 40–48. DOI: https://doi.org/10.1109/CVPRW.2019.00011

[97] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. 2011. Reading digits in natural images with unsupervised feature learning. (2011).

[98] Pascal Paillier. 1999. Public-key cryptosystems based on composite degree residuosity classes. *Public-key Cryptosyst. Based Compos. Degree Residuos. Class.* 5 (1999), 223–238. DOI: https://doi.org/10.1007/3-540-48910-X_16

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

[100] Nicolas Papernot, Shuang Song, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Úlfar Erlingsson. 2018. Scalable private learning with PATE. *arXiv preprint arXiv:1802.08908* (2018).

[101] Dario Pasquini, Giuseppe Ateniese, and Massimo Bernaschi. 2021. Unleashing the tiger: Inference attacks on split learning. In *ACM SIGSAC Conference on Computer and Communications Security.* 2113–2129.

[102] Arpita Patra and Ajith Suresh. 2020. BLAZE: Blazing fast privacy-preserving machine learning. *CoRR* abs/2005.09042 (2020).

[103] Zhe Peng, Jianliang Xu, Xiaowen Chu, Shang Gao, Yuan Yao, Rong Gu, and Yuzhe Tang. 2022. VFChain: Enabling verifiable and auditable federated learning via blockchain systems. *IEEE Trans. Netw. Sci. Eng.* 9, 1 (2022), 173–186. DOI: https://doi.org/10.1109/TNSE.2021.3050781

[104] George-Liviu Pereteanu, Amir Alansary, and Jonathan Passerat-Palmbach. 2022. Split HE: Fast secure inference combining split learning and homomorphic encryption. *arXiv preprint arXiv:2202.13351* (2022).

[105] Le Trieu Phong, Yoshinori Aono, Takuya Hayashi, Lihua Wang, and Shiho Moriai. 2018. Privacy-preserving deep learning via additively homomorphic encryption. *IEEE Trans. Inf. Forens. Secur.* 13, 5 (2018), 1333–1345. DOI: https://doi.org/10.1109/TIFS.2017.2787987

[106] Tran Thi Phuong et al. 2019. Privacy-preserving deep learning via weight transmission. *IEEE Trans. Inf. Forens. Secur.* 14, 11 (2019), 3003–3015.

[107] Youyang Qu, Mohammad Reza Nosouhi, Lei Cui, and Shui Yu. 2021. Leading attacks in privacy protection domain. In *Personalized Privacy Protection in Big Data*. Springer, 15–21.

[108] Michael Rabin. 2005. How to exchange secrets with oblivious transfer. *IACR Cryptology ePrint Archive* 2005 (01 2005), 187.

[109] Rahul Rachuri and Ajith Suresh. 2019. Trident: Efficient 4PC framework for privacy preserving machine learning. *CoRR* abs/1912.02631 (2019).

[110] Brandon Reagen, Wooseok Choi, Yeongil Ko, Vincent Lee, Gu-Yeon Wei, Hsien-Hsin S. Lee, and David Brooks. 2020. Cheetah: Optimizing and accelerating homomorphic encryption for private inference. DOI: https://doi.org/10.48550/ARXIV.2006.00505

[111] Alessandro Renda, Pietro Ducange, Francesco Marcelloni, Dario Sabella, Miltiadis C. Filippou, Giovanni Nardini, Giovanni Stea, Antonio Virdis, Davide Micheli, Damiano Rapone, et al. 2022. Federated learning of explainable AI models in 6G systems: Towards secure and automated vehicle networking. *Information* 13, 8 (2022), 395.

[112] M. Riazi, Mohammad Samragh, Hao Chen, K. Laine, K. Lauter, and F. Koushanfar. 2019. XONN: XNOR-based oblivious deep neural network inference. In *IACR Cryptol. ePrint Arch.*

[113] Sadegh Riazi, Christian Weinert, Oleksandr Tkachenko, Ebrahim M. Songhori, Thomas Schneider, and Farinaz Koushanfar. 2018. Chameleon: A hybrid secure computation framework for machine learning applications. (01 2018).

[114] Maria Rigaki and Sebastian Garcia. 2023. A survey of privacy attacks in machine learning. *Comput. Surv.* 56, 4 (2023), 1–34.

[115] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. 1978. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM* 21, 2 (1978), 120–126.

[116] Bita Darvish Rouhani, M. Sadegh Riazi, and Farinaz Koushanfar. 2018. DeepSecure: Scalable provably-secure deep learning. In *Design Automation Conference (DAC'18)*, Vol. Part F1377. 1–6. arXiv:1705.08963

[117] Theo Ryffel, Pierre Tholoniat, David Pointcheval, and Francis R. Bach. 2022. AriaNN: Low-interaction privacy-preserving deep learning via function secret sharing. *Proc. Privac Enhanc. Technol.* 2022 (2022), 291–316.

[118] C. Okan Sakar, Gorkem Serbes, Aysegul Gunduz, Hunkar C. Tunc, Hatice Nizam, Betul Erdogdu Sakar, Melih Tutuncu, Tarkan Aydin, M. Erdem Isenkul, and Hulya Apaydin. 2019. A comparative analysis of speech signal processing algorithms for Parkinson's disease classification and the use of the tunable Q-factor wavelet transform. *Appl. Soft Comput.* 74 (2019), 255–263.

[119] Amartya Sanyal, Matt J. Kusner, Adrià Gascón, and Varun Kanade. 2018. TAPAS: Tricks to accelerate (encrypted) prediction as a service. *CoRR* abs/1806.03461 (2018).

[120] Sinem Sav, Apostolos Pyrgelis, Juan Ramón Troncoso-Pastoriza, David Froelicher, Jean-Philippe Bossuat, Joao Sa Sousa, and Jean-Pierre Hubaux. 2020. POSEIDON: Privacy-preserving federated neural network learning. *CoRR* abs/2009.00349 (2020).

[121] SEAL 2018. Microsoft SEAL (release 3.0). Microsoft Research, Redmond, WA. Retrieved from http://sealcrypto.org.

[122] Mohamed Seif, Ravi Tandon, and Ming Li. 2020. Wireless federated learning with local differential privacy. *arXiv preprint arXiv:2002.05151* (2020).

[123] Adi Shamir. 1979. How to share a secret. *Commun. ACM* 22, 11 (Nov. 1979), 612–613. DOI: https://doi.org/10.1145/359168.359176

[124] Reza Shokri and Vitaly Shmatikov. 2015. Privacy-preserving deep learning. In *22nd ACM SIGSAC Conference on Computer and Communications Security*. 1310–1321.

[125] Jinhyun So, Başak Güler, and A. Salman Avestimehr. 2021. Turbo-aggregate: Breaking the quadratic aggregation barrier in secure federated learning. *IEEE J. Select. Areas Inf. Theor.* 2, 1 (2021), 479–489. DOI: https://doi.org/10.1109/JSAIT.2021.3054610

[126] Beata Strack, Jonathan P. DeShazo, Chris Gennings, Juan L. Olmo, Sebastian Ventura, Krzysztof J. Cios, and John N. Clore. 2014. Impact of HbA1c measurement on hospital readmission rates: Analysis of 70,000 clinical database patient records. *BioMed Res. Int.* 2014 (2014).

[127] Lichao Sun, Jianwei Qian, Xun Chen, and Philip S. Yu. 2020. *LDP-FL: Practical Private Aggregation in Federated Learning with Local Differential Privacy*. Technical Report. arXiv:2007.15789v1

[128] Fengyi Tang, Wei Wu, Jian Liu, Huimei Wang, and Ming Xian. 2019. Privacy-preserving distributed deep learning via homomorphic re-encryption. *Electronics* 8, 4 (2019). DOI: https://doi.org/10.3390/electronics8040411

[129] Harry Chandra Tanuwidjaja, Rakyong Choi, Seunggeun Baek, and Kwangjo Kim. 2020. Privacy-preserving deep learning on machine learning as a service—A comprehensive survey. *IEEE Access* 8 (2020), 167425–167447. DOI: https://doi.org/10.1109/ACCESS.2020.3023084

[130] Tensorflow. 2019. Text Generation with an RNN. Retrieved from https://www.tensorflow.org/tutorials/text/text_generation

[131] Chandra Thapa, Pathum Chamikara Mahawaga Arachchige, Seyit Camtepe, and Lichao Sun. 2022. SplitFed: When federated learning meets split learning. In *AAAI Conference on Artificial Intelligence*, Vol. 36. 8485–8493.

[132] Stacey Truex, Nathalie Baracaldo, Ali Anwar, Thomas Steinke, Heiko Ludwig, Rui Zhang, and Yi Zhou. 2019. A hybrid approach to privacy-preserving federated learning. In *12th ACM Workshop on Artificial Intelligence and Security*. 1–11.
[133] Valeria Turina, Zongshun Zhang, Flavio Esposito, and Ibrahim Matta. 2021. Federated or split? A performance and privacy analysis of hybrid split and federated learning architectures. In *IEEE 14th International Conference on Cloud Computing (CLOUD'21)*. IEEE, 250–260.
[134] Praneeth Vepakomma, Otkrist Gupta, Tristan Swedish, and Ramesh Raskar. 2018. Split learning for health: Distributed deep learning without sharing raw patient data. *arXiv preprint arXiv:1812.00564* (2018).
[135] Sameer Wagh, Divya Gupta, and Nishanth Chandran. 2018. SecureNN: Efficient and private neural network training. *IACR Cryptol. ePrint Arch.* 2018 (2018), 442.
[136] Sameer Wagh, Divya Gupta, and Nishanth Chandran. 2019. SecureNN: Efficient and private neural network training. In *Privacy Enhancing Technologies Symposium (PETS'19)*. Retrieved from https://www.microsoft.com/en-us/research/publication/securenn-efficient-and-private-neural-network-training/
[137] Sameer Wagh, Shruti Tople, Fabrice Benhamouda, Eyal Kushilevitz, Prateek Mittal, and Tal Rabin. 2020. FALCON: Honest-majority maliciously secure framework for private deep learning. *CoRR* abs/2004.02229 (2020).
[138] Hua Wang, Sheng Gao, Huanyu Zhang, Milan Shen, and Weijie J. Su. 2022. Analytical composition of differential privacy via the Edgeworth Accountant. *arXiv preprint arXiv:2206.04236* (2022).
[139] Yansheng Wang, Yongxin Tong, and Dingyuan Shi. 2020. *Federated Latent Dirichlet Allocation: A Local Differential Privacy Based Framework*. Technical Report. Retrieved from www.aaai.org
[140] Yue Wang, Xintao Wu, and Donghui Hu. 2016. Using randomized response for differential privacy preserving data collection. In *EDBT/ICDT Workshops*, Vol. 1558.
[141] Stanley L. Warner. 1965. Randomized response: A survey technique for eliminating evasive answer bias. *J. Am. Stat. Assoc.* 60, 309 (1965), 63–69.
[142] Wenqi Wei, Ling Liu, Yanzhao Wut, Gong Su, and Arun Iyengar. 2021. Gradient-leakage resilient federated learning. In *IEEE 41st International Conference on Distributed Computing Systems (ICDCS'21)*. IEEE, 797–807.
[143] Guowen Xu, Hongwei Li, Sen Liu, Kan Yang, and Xiaodong Lin. 2019. VerifyNet: Secure and verifiable federated learning. *IEEE Trans. Inf. Forens. Secur.* 15 (2019), 911–926.
[144] Guowen Xu, Hongwei Li, Sen Liu, Kan Yang, and Xiaodong Lin. 2020. VerifyNet: Secure and verifiable federated learning. *Trans. Inf. Forens. Secur.* 15 (Jan. 2020), 911–926. DOI: https://doi.org/10.1109/TIFS.2019.2929409
[145] Guowen Xu, Hongwei Li, Yun Zhang, Shengmin Xu, Jianting Ning, and Robert H. Deng. 2022. Privacy-preserving federated deep learning with irregular users. *IEEE Trans. Depend. Secure Comput.* 19, 2 (2022), 1364–1381. DOI: https://doi.org/10.1109/TDSC.2020.3005909
[146] Runhua Xu, Nathalie Baracaldo, Yi Zhou, Ali Anwar, and Heiko Ludwig. 2019. HybridAlpha: An efficient approach for privacy-preserving federated learning. *CoRR* abs/1912.05897 (2019).
[147] Yuan Xu. 2004. On discrete orthogonal polynomials of several variables. *Advan. Appl. Math.* 33, 3 (2004), 615–632. DOI: https://doi.org/10.1016/j.aam.2004.03.002
[148] Yushuang Yan, Qingqi Pei, and Hongning Li. 2019. Privacy-preserving compressive model for enhanced deep-learning-based service provision system in edge computing. *IEEE Access* 7 (2019), 92921–92937.
[149] Andrew Chi-Chih Yao. 1986. How to generate and exchange secrets. In *27th Annual Symposium on Foundations of Computer Science (SFCS'86)*. 162–167. DOI: https://doi.org/10.1109/SFCS.1986.25
[150] Xuefei Yin, Yanming Zhu, and Jiankun Hu. 2021. A comprehensive survey of privacy-preserving federated learning: A taxonomy, review, and future directions. *ACM Comput. Surv.* 54, 6, Article 131 (July 2021), 36 pages. DOI: https://doi.org/10.1145/3460427
[151] Zhengxin Yu, Jia Hu, Geyong Min, Zi Wang, Wang Miao, and Shancang Li. 2021. Privacy-preserving federated deep learning for cooperative hierarchical caching in fog computing. *IEEE Internet Things J.* (2021).
[152] Sen Yuan, Milan Shen, Ilya Mironov, and Anderson C. A. Nascimento. 2021. Practical, label private deep learning training based on secure multiparty computation and differential privacy. *Cryptology ePrint Archive* (2021).
[153] Chengliang Zhang, Suyi Li, Junzhe Xia, Wei Wang, Feng Yan, and Yang Liu. 2020. BatchCrypt: Efficient homomorphic encryption for cross-silo federated learning. In *USENIX Annual Technical Conference (USENIX ATC'20)*.
[154] Jiale Zhang, Junyu Wang, Yanchao Zhao, and Bing Chen. 2019. An efficient federated learning scheme with differential privacy in mobile edge computing. In *International Conference on Machine Learning and Intelligent Communications*. Springer, 538–550.
[155] Jun Zhang, Zhenjie Zhang, Xiaokui Xiao, Yin Yang, and Marianne Winslett. 2012. Functional mechanism: Regression analysis under differential privacy. *arXiv preprint arXiv:1208.0219* (2012).
[156] Qiao Zhang, Cong Wang, Hongyi Wu, Chunsheng Xin, and T. V. X. Phuong. 2018. GELU-Net: A globally encrypted, locally unencrypted deep neural network for privacy-preserved learning. 3933–3939. DOI: https://doi.org/10.24963/ijcai.2018/547

[157] Qiao Zhang, Chunsheng Xin, and Hongyi Wu. 2021. Privacy-preserving deep learning based on multiparty secure computation: A survey. *IEEE Internet Things J.* 8, 13 (2021), 10412–10429. DOI:https://doi.org/10.1109/JIOT.2021.3058638

[158] Lingchen Zhao, Qian Wang, Qin Zou, Yan Zhang, and Yanjiao Chen. 2019. Privacy-preserving collaborative deep learning with unreliable participants. *IEEE Trans. Inf. Forens. Secur.* 15 (2019), 1486–1500.

[159] Qinqing Zheng, Shuxiao Chen, Qi Long, and Weijie J. Su. 2021. *Federated f-Differential Privacy.* Technical Report. arXiv:2102.11158v1

[160] Jun Zhou, Zhenfu Cao, Xiaolei Dong, and Xiaodong Lin. 2015. PPDM: A privacy-preserving protocol for cloud-assisted e-healthcare systems. *IEEE J. Select. Topics Signal Process.* 9, 7 (2015), 1332–1344.

[161] Junhao Zhou, Yufei Chen, Chao Shen, and Yang Zhang. 2021. Property inference attacks against GANs. *arXiv preprint arXiv:2111.07608* (2021).

[162] Hangyu Zhu, Rui Wang, Yaochu Jin, Kaitai Liang, and Jianting Ning. 2021. Distributed additive encryption and quantization for privacy preserving federated deep learning. *Neurocomputing* 463 (2021), 309–327. DOI:https://doi.org/10.1016/j.neucom.2021.08.062

[163] Ruozhou Zuo, Haibo Tian, Zhiyuan An, and Fangguo Zhang. 2022. Post-quantum privacy-preserving aggregation in federated learning based on lattice. In *Cyberspace Safety and Security: 14th International Symposium, CSS 2022, Xi'an, China, October 16–18, 2022, Proceedings*. Springer, 314–326.

Received 7 April 2023; revised 27 April 2024; accepted 19 October 2024