Received 9 August 2022, accepted 24 August 2022, date of publication 5 September 2022, date of current version 20 September 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3204037

SURVEY

A Survey and Guideline on Privacy Enhancing Technologies for Collaborative Machine Learning

ELIF USTUNDAG SOYKAN, LEYLI KARAÇAY, FERHAT KARAKOÇ, AND EMRAH TOMUR
Ericsson Research, 34390 Istanbul, Türkiye

Corresponding author: Elif Ustundag Soykan (elif.ustundag.soykan@ericsson.com)

This work was supported in part by The Scientific and Technological Research Council of Turkey (TUBITAK) through the 1515 Frontier Research and Development Laboratories Support Program under Project 5169902, and in part by the European Commission through the H2020 Project Hexa-X under Grant 101015956.

ABSTRACT As machine learning and artificial intelligence (ML/AI) are becoming more popular and advanced, there is a wish to turn sensitive data into valuable information via ML/AI techniques revealing only data that is allowed by concerned parties or without revealing any information about the data to third parties. Collaborative ML approaches like federated learning (FL) help tackle these needs and concerns, bringing a way to use sensitive data without disclosing critically sensitive features of that data. In this paper, we provide a detailed analysis of state of the art for collaborative ML approaches from a privacy perspective. A detailed threat model and security and privacy considerations are given for each collaborative method. We deeply analyze Privacy Enhancing Technologies (PETs), covering secure multi-party computation (SMPC), homomorphic encryption (HE), differential privacy (DP), and confidential computing (CC) in the context of collaborative ML. We introduce a guideline on the selection of the privacy preserving technologies for collaborative ML and privacy practitioners. This study constitutes the first survey to provide an in-depth focus on collaborative ML requirements and constraints for privacy solutions while also providing guidelines on the selection of PETs.

INDEX TERMS Collaborative machine learning, privacy enhancing technologies, privacy, security, federated learning, artificial intelligence, ML, AI.

I. INTRODUCTION

Machine Learning (ML) and Artificial Intelligence (AI) techniques are becoming popular for realizing autonomous applications and for benefiting from data in a clever way [1]. ML helps organizations to extract meaningful conclusions from their data. For some cases, like medical studies, having sufficiently large and diverse data sets is the key challenge. As the data is the fuel of ML, having more data paves the way for high-quality models. On the other hand, the required amount and quality of training data may not be owned by any single party in a collaborative ML process. A naive approach to generate high quality models would entail collecting data from multiple parties to train a global model. However, sharing large quantities of data introduces communication inefficiencies, while exposing sensitive data to external parties for the training process may also pose significant commercial, privacy, and legal risks. Collaborative ML approaches help to tackle these needs and concerns, introducing a way to use multiple parties' data without forcing parties to share sensitive data.

Collaborative ML approaches differ in their approaches to handling data and models, along with their trust assumptions between parties. Parties can collaborate by sharing their data, or splitting the model training computation, or both. For data collaboration, Federated Learning (FL) is one of the most adopted methodologies where a central model aggregator is needed. Split Learning (SL) employs the trust relation similar to FL, but it focuses on splitting the model into different parts. Decentralized learning (DecL) approaches remove the central aggregator requirement by involving each party in model training individually and distributing the partial model

The associate editor coordinating the review of this manuscript and approving it for publication was Yeliz Karaca.

to achieve a common trained model. All these methodologies can be considered as distributed learning cases. In addition, distributed learning (DistL) also addresses those approaches that focus more on efficient use of computational resources in the presence of big data or large-scale modeling. Recently, some studies focus on developing efficient enablers for distributed learning e.g., matrix factorization techniques, distributed alternating direction method of multipliers [2], [3]. Some of those works shed light on privacy concerns [4]. In our work we largely omit DistL, as it requires substantially different analysis and domain knowledge.

The most important driver in the emergence of collaborative learning models is privacy. These techniques are developed in which privacy-aware approaches as training data is not shared among the collaborating parties, but it is arguable that existing solutions meet all the privacy requirements. The application of Privacy Enhancing Technologies (PETs) for ML represents an approach to address the remaining privacy concerns where there is a wish to turn sensitive data into valuable information via ML techniques revealing only data allowed by concerned parties or without revealing any information about the data to the third parties. PETs constitute a set of methods that can be used to improve privacy containing different building blocks including secure multi-party computation (SMPC), homomorphic encryption (HE), differential privacy (DP), confidential computing using trusted execution environments (TEE) [5]. While selecting a method from this set, a practitioner must take several elements into account, including collaboration topology with ML algorithm, the threat model, and the use case constraints on computation and communication. Thus, a wide perspective is required to decide on the best possible privacy-enhanced solution. In this study, we concentrate on PETs in collaborative ML settings and present a survey and a guideline on the choice of PET solution in this area. For this aim, we first present an analysis of collaborative ML model techniques with their requirements and constraints and list security and privacy-related attacks against them. Then, after giving some brief information about the PET solutions, we provide our guideline on the selection of the PET method to be used in the collaborative ML technique.

The main contributions of this work are as follows:

- Different collaborative ML (federated learning, split learning, decentralized learning) techniques are elaborated from an architectural and privacy perspectives.
- Security and privacy considerations are explained for each collaborative technique, and a threat model is provided.
- A comprehensive review of existing works on collaborative ML privacy by focusing on the use of privacy-enhancing technologies is presented.
- Collaborative ML requirements and constraints are presented with privacy-enhancing solutions scope by proposing a how-to methodology for ML and privacy practitioners. This is the first work to shed light on the PET selection process to the best of our knowledge.

The rest of the paper continues as follows. First, some background information about privacy objectives for ML is provided in section II. Then related survey works are given in section III. Section IV covers different collaborative ML techniques. Potential privacy and security and attacks on collaborative ML techniques and threat models are provided in section V. PETs and current applications to prevent privacy attacks in collaborative ML are presented in section VI. In section VII, we propose a methodology on how PETs should be chosen under different constraints and requirements. Section VIII provides a discussion on open issues and future directions. The paper concludes in section IX.

II. PRIVACY OBJECTIVES FOR ML

ML itself is about understanding data and benefiting from it. When we say data, we refer to a broad variety of data sources, including (but not necessarily limited to) application specific data, telemetry data, behavioral data or personally identifiable information (PII). In the ML context, data can be used during training and inference phases. Data content may include private indicators explicitly like PII that require special care. Even if the data does not include PII, it may be possible to extract sensitive information from the data if it includes potentially linkable or interpretable auxiliary information. Studies [6], [7], [8] unveil that model parameters are as important as data since they may leak information about the processed data. Therefore, where the data is kept and used is as important as its content. During the ML life cycle, which comprises training, model creation, and model inference, privacy issues may exist depending on the threat and data ownership model. We generalize privacy concerns into the following three categories:

1) PRIVACY FOR TRAINING DATA

Privacy breaches for training data may occur via unauthorized direct access to the data. Training data exposure may also happen in Machine Learning as a Service (MLaaS) setting, where the model can be trained on the cloud. Monitoring by the data owner may not be possible if the cloud service does not provide any auditing mechanism on how and where the data is processed and stored during training and how it is erased after the training. In addition to protecting training data from unauthorized access, practitioners may want to protect the data against third parties who utilize the data to create an ML model.

2) PRIVACY FOR MODEL INFERENCE

Even though the data or models are isolated from the adversary, vulnerabilities remain during the inference phase. In this phase, there can be two different privacy concerns: 1) protection of the model parameters, and 2) protection of query data. In the first case, the adversary may collect the query/result pairs and try to recover the model parameters. The transferability property of many models and memory of deep neural networks (DNNs) enables this kind of attack [6]. For the second type of privacy threat, the user of the model, who sends queries to the model, may want to learn query result without revealing any information about the query itself.

3) PROTECTION OF MODEL AT REST AND IN TRANSIT

Trained models can be considered as intellectual property of model owners. Hence, accessing or extracting information about a model can lead to negative business implications. Model hyper-parameters, trained model parameters, or other generic information about the model can be extracted via gaining access to shared environments e.g., from a Machine Learning as a Service (MLaaS) setting [9]. Also, the model itself and the local model updates computed by clients in some collaborative ML settings such as FL need security and privacy protection while they are in transit in addition to protection at rest.

III. EXISTING SURVEYS

In this section, existing surveys related to this study are investigated according to their scope and relation to our work. We focus on the surveys in the collaborative ML/AI privacy domain.

In [10], Mothukuri *et al.* presented security and privacy aspects of FL from different perspectives. They provided available approaches and technologies by means of network topology, data availability (cross-silo/cross-device), data partitioning (horizontal, vertical, transfer learning), aggregation algorithms (FedAvg [22], etc.), and open-source frameworks (like Tensorflow [23]). Differently from our approach, they classified decentralized ML under FL as one of the network topologies. On the other hand, the work did not explicitly describe the security and privacy aspects of decentralized ML. Mothukuri *et al.* also included a severity analysis, in their work, for security attacks based on the probability of the adversary exploiting vulnerabilities resulted from possible different sources. Although the severity evaluation was not clear, the analysis depicted where most of the attacks are concentrated. Distributed ML specific attacks were out of their scope like in our work, but still, the work highlights the common attacks on FL and distributed ML. For privacy solutions, the study covered the SMPC and DP as PETs. Two additional mitigation strategies, VerifyNet [24] and Adversarial Training, were presented in that work as inference phase protections where VerifyNet introduces a double-masking protocol and Adversarial Training adds adversarial samples to minimize the leakage for the latter.

Blanco-Justicia *et al.* [11] pointed out both security and privacy attacks against federated learning. The authors analyzed these types of attacks and surveyed existing solutions against them. According to the survey, the privacy solutions make it difficult to prevent security attacks such as poisoning attacks because the server may not be able to analyze the updates sent encrypted by the client using privacy enhancing solutions. The survey states the construction of solutions that can mitigate the security and privacy attacks at the same time as a research challenge. Also, the survey proposes a possible way forward to construct such solutions. According to the proposal, anonymizing the sender of the model updates is enough to mitigate privacy attacks while enabling possible integration of solutions against security attacks. Since the sender will be known by neither the server nor the other clients, there will be no privacy threat and since there will be no encryption requirement for the updates sent by the clients to the server, the server can analyze the updates to prevent security attacks. To hide the identity of the sender client, the survey proposes a peer-to-peer privacy-enhanced data forwarding solution.

Boulemtafes *et al.* [12] provided a multi-level taxonomy from a deep learning perspective. The study considered the related works by dividing them into three ML phases; learning, inference, and release. For each phase, they investigated existing works with a different multi-level classification providing performance metrics. For the learning phase taxonomy, they presented works both for collaborative and traditional ML settings. Then each setting was analyzed in two categories as server-based and server-assisted. In the server-based category, all training is done on servers while in the server-assisted setting the training is performed collaboratively like in FL. In the inference phase, server-based and server-assisted PET studies were given. However, collaborative and traditional ML settings were not considered in this phase. For the model release phase, only differential privacy techniques were considered. The usage of confidential computing technologies is not considered in the study and privacy preserving collaborative ML studies were covered only under the learning phase, not under the inference and the release phases.

Li *et al.* [13] provided a taxonomy for federated learning, which classifies FL considering data distribution, machine learning model, privacy mechanism, communication architecture, the scale of federation, and motivation of federation. The authors analyzed mostly FL system building blocks and presented a comparison with conventional federated database and cloud systems. Existing studies were summarized with different aspects without dealing with the issue of privacy in detail. A design guideline was provided for FL considering effectiveness, efficiency, privacy, and autonomy as the design factors. Although their perspective resembles ours, our guideline focuses more deeply on privacy aspects.

Yin *et al.* [14] provided a 5W-based (who, what, when, where, why) taxonomy of privacy leakages in FL. The authors presented state of the art privacy preserving solutions covering HE, SMPC, DP, and other perturbation techniques. Solutions using trusted execution environment are not considered. Our study differentiates from that study by considering other collaborative ML techniques, including trusted execution environments as a privacy enhancing tool, comparing the PETs and providing a guideline.

In [15], Kairouz *et al.* provided a survey with a comprehensive description of FL and challenges considering the attack strategies for cross-device settings. The study targeted FL and considered decentralized learning, cross-silo FL, and split learning as a relaxation of FL. Although their focus was not

TABLE 1. **Summary of existing surveys.**

Reference	Year	Privacy Focus	Collaboration Setting	PETs Covered	# of References
Mothukuri et al. [10]	2021	Attacks and Defences	FL and DecL	SMPC, DP	215
Blanco-Justicia et al. [11]	2021	Attacks and Defences	FL	SMPC, DP	37
Boulemtafes et al. [12]	2020	Solutions	FL	SMPC, DP, HE	73
Li et al. [13]	2021	Solutions	FL	SMPC, DP, TEE	228
Yin et al. [14]	2021	Attacks and Defences	FL	SMPC, DP, HE	229
Kayrouz et al. [15]	2021	Attacks and Defences	FL	SMPC, DP, HE, TEE	511
Zhang et al. [16]	2018	Solutions	FL	SMPC, DP, HE, TEE	55
Ma et al. [17]	2020	Challenges	FL	SMPC, DP, HE	15
Vepakomma et al. [18]	2018	Solutions	FL and SL	SMPC, DP, HE	115
Rigaki et al. [19]	2020	Attacks	FL and DistL	DP	131
Lyu et al. [20]	2020	Attacks	FL and DecL	DP, HE	43
Enthoven et al. [21]	2021	Attacks	FL	SMPC, DP, HE	81
Our survey	2022	Attacks and Defences	FL, DecL, SL	SMPC, DP, HE, TEE	155

solely on privacy, the authors presented privacy technologies for user data from three perspectives: how FL computation is done, what is computed by FL algorithm, and the execution verification of the computed model. The addressed privacy technologies were given as SMPC, HE, DP, and TEE.

In [16], Zhang *et al.* presented a short survey for deep learning focusing on two settings: direct collaborative learning, where user data is aggregated on central servers, and indirect collaborative learning such as FL setting. Their privacy scope addressed direct and indirect learning cases by investigating some works that use HE, TEE, SMPC, and DP. Ma *et al.* [17] pointed out security and privacy issues in FL by concentrating on convergence, poisoning, model aggregation, and scale-up challenges in designing security and privacy in FL solutions.

Split learning privacy aspects were covered in [18]by Vepakomma *et al.* along with the FL and large batch synchronous stochastic gradient descent (SGD) for deep learning. The study provides comparisons for these three techniques by means of the privacy level offered over data, hyper-parameters, intermediate model representations, and resource requirements such as computation, memory, and bandwidth without considering any privacy enhancing methods. Further, they suggested the use of DP, SMPC, and HE techniques to enhance the privacy level.

Some studies [19], [20], [21] focused on privacy attacks aiming to reconstruct training samples or inferring properties by providing attack taxonomies. These studies also included defense strategies to address the attacks by using PETs and other techniques such as changing the collaborative learning setting like selective sharing of gradients, compressing the gradients, robust aggregation, or employing the dropouts. However, these techniques do not provide as strong of privacy guarantees as PETs do.

Previous surveys consider FL challenges and issues with different aspects providing privacy perspective at some level. In comparison with the previous works, our study aims to investigate all aspects of privacy attacks and solutions covering broader set of PETs for not only FL but also decentralized learning and split learning. In addition, we aim to support practitioners by giving a guideline on selecting the most appropriate privacy enhancing technologies based on their needs and the collaborative setting.

IV. COLLABORATIVE ML

In recent years, collaborative utilization of the distributed data owned by different data owners is in great demand since data is distributed among different entities. Depending on the application scenarios, collaborative learning can be generally classified as cross-device and cross-silo learning. In cross-device learning, the clients are mobile or IoT devices having limited computing power and possibly unreliable communication. In contrast, the clients in cross-silo learning settings are a small number of organizations (e.g., medical) with reliable communications. It is important to understand the core challenges of the collaborative ML settings such as communication, computation, cost and privacy requirements to design efficient models. In this section, we provide background information on collaborative ML models. We consider Federated Learning, Decentralized Learning, and Split Learning as collaboration methods. There can be other methods where parties can collaborate, but for generalization and common understanding, we limit the study to these three methods. Centralized learning may also be considered a collaboration technique since the data is shared by end devices to contribute to the global model. On the other hand, there is no decoupling during the model training in this setting. For the sake of completeness, we provide a definition of this model, but it is not considered in the privacy discussions for the rest of the paper.

A. CENTRALIZED LEARNING

In centralized learning, illustrated in Figure 1, each client transfers its own data to the server, where data is aggregated, and then the model training is performed centrally. As a result, one single model is produced, which can be made available to the clients via either sending model to the clients or enable inference over the server. In this kind of setting, the server usually has more computing power than clients, e.g., the server is located in the cloud or has access to larger computing resources. Although this gives computation flexibility, it may not be preferred when communication cost is a concern as all data is transferred to the server. Additionally, data transfer may not be allowed due to user privacy or legislative reasons.

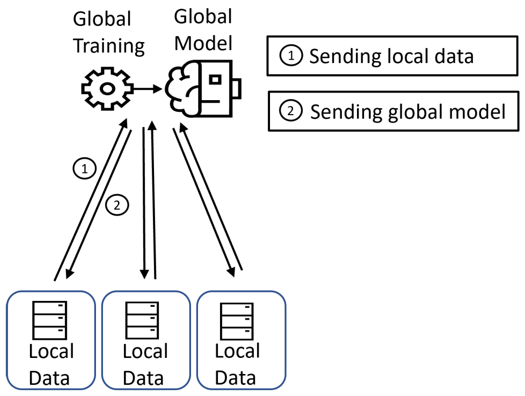

FIGURE 1. Centralized Learning Topology.

B. FEDERATED LEARNING

FL [22] is a collaborative ML technique that allows data owners to jointly train a model with the help of a server without revealing the data to other data owners and the server. The main steps in this technique are that a trusted server initializes the training and sends the initial model to data owners. Each data owner locally trains the model using their local sensitive data. Only updated parameters are sent to the server for global model aggregation. These steps are repeated until the training is completed. The overall architecture of FL is depicted in Figure 2.

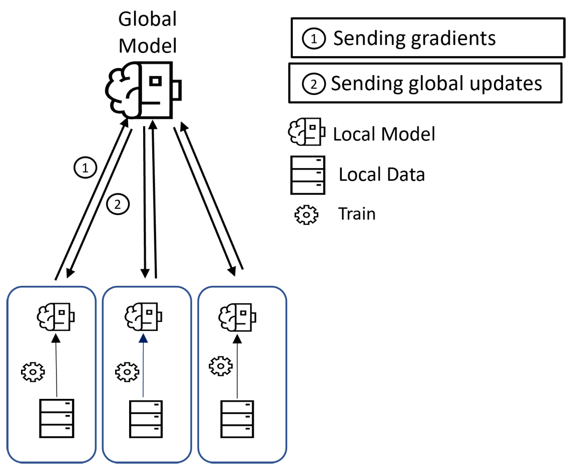

FIGURE 2. Federated Learning Topology.

Only the server is allowed to update the parameters in the global model. One example case is Google Keyboard [25] which includes features like text auto-correction, word completion, and next word or emoji prediction. Without collecting sensitive raw data from users, the models are trained collaboratively using FL. The result computed by the server is returned to each user. A principal advantage of this approach is that there is no need to directly access raw training data for model training.

FL can be classified as horizontal FL (HFL), vertical FL (VFL), and federated transfer learning (FTL) with respect to how data is distributed over a sample or feature spaces among clients [26]. In HFL, the samples are different for each data owner, but they share the same feature space. For this scenario, a server can aggregate the information from different data owners. In VFL, there is a large overlap in the sample spaces among multiple clients, but the feature spaces are different. A variety of secure models are proposed for VFL, including association rule mining, decision tree, and Naïve Bayes classifier. In [27], the authors propose a secure machine learning where data is partitioned in feature space. FL can be implemented using the most popular ML algorithms such as neural networks, decision tree, and linear/logistic regression where data is horizontally partitioned [13]. In the case of vertical FL, a more complex mechanism to decompose the loss function at each party is required [28], [29]. Being in the same feature space and having the same distribution are key assumptions for training and testing data in many machine learning algorithms. However, in many real-world applications, we may have two or more domains of interest where users' data have different feature spaces and follow a different data distribution. FTL is a new learning framework emerging in recent years where the data has little overlap over the sample and feature space. Benefiting from transfer learning [30] where knowledge (features and weights) from previously trained models is used for training newer models, FTL brings solutions for both the sample and feature space. Using FTL, the complementary knowledge is transferred across domains in a federation; thus, flexible and effective models can be built for the target domain using the information from the other source domains.

FL approaches may differ in optimization strategies that are performed during training cycles. The most preferred optimization method is SGD. FL approaches can use SGD in different settings. Federated SGD (FedSGD) and Federated Averaging (FedAVG) are the two algorithms that are widely adopted in the FL implementations [22]. FedSGD implements a single batch gradient calculation on the local model. In each round, clients perform the gradient calculation for a single batch and send the result to the server. Then the server aggregates the gradient updates from each client then applies the update. However, this requires a large number of rounds to achieve the desired convergence. FedAVG suggests an improvement to reduce the number of rounds by increasing the number of gradient calculations on the local model. In FedAVG, each client performs the iteration multiple times (called epoch number) on the gradient. Then, the locally updated weights are sent to the server.

While the training data is not shared with the server or with the other data owners, there are still privacy concerns about the deployment of federated learning because the model updates shared with the server may reveal sensitive information about training data. Therefore, a number of works try to address this issue [31], [32], [33].

C. DECENTRALIZED LEARNING

In decentralized learning (DecL), the aim is that nodes have peer-to-peer communication with each other instead of having communication with the server, where each node has its own data and performs its own learning. A connected graph represents the communication topology as illustrated in Figure 3 where the communication channel between two nodes is indicated by an arrow. Each node sends/receives messages to/from a certain number of peers. Each node performs a local update in each round of a fully decentralized algorithm, and updates are exchanged with neighbors. The global state of the model is not iterated by the server as in FL, but the process proceeds such that individual models converge to the desired global model. In DecL, setting up a learning task (such as deciding on the algorithm to be used, the hyper-parameters) might be done by a central authority who can alternatively be a node or through a consensus scheme in a collaborative way [15]. In this learning model, no centralized infrastructure is needed, and data is fully distributed. As in the FL case, privacy concerns still remain although the model provides better data privacy compared to centralized topology since the models are shared instead of participants' sensitive data.

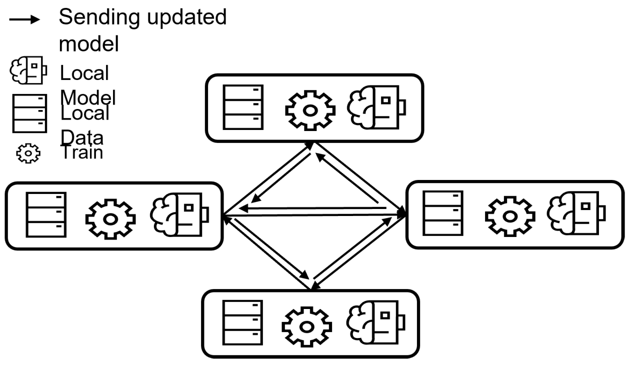

FIGURE 3. **Decentralized Learning Topology.**

D. SPLIT LEARNING

Split learning (SL) is a type of distributed deep learning method. It allows different cooperative clients to train deep learning models without sharing any training data or detailed information about the model [34]. In a split learning setting, each client trains a deep neural network up to a specific layer called the cut layer. The rest of the flow is completed by sending the outputs at the cut layer to a server without observing raw data from any client that holds it, as shown in Figure 4. In this way, without sharing raw data, a round of forward propagation is being completed. Then, at the server, the gradients are back propagated until the cut layer. The gradients at the cut layer are sent back to clients. Until the learning task is converged, this process is continued. In split learning, the activations and gradients are communicated just from the split layer, unlike other methods where the parameters resulting from local training tasks are shared. There is a difference between split learning and federated learning regarding computation overload on the clients and how quickly split learning converges. Also, in terms of communication bandwidth, when the number of clients or the model size is large, split learning is more communication efficient. If the training data size is large but the number of clients and model size is small, FL is more communication efficient [35].

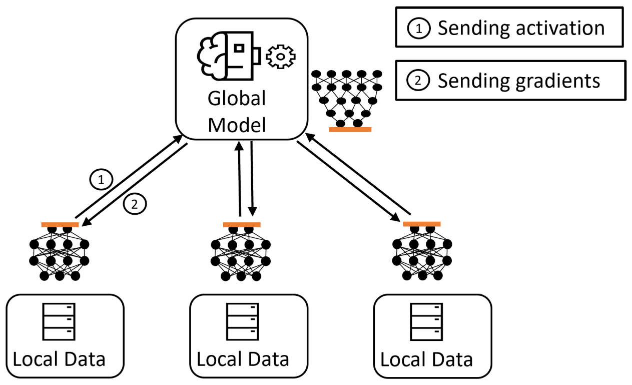

FIGURE 4. **Split Learning Topology.**

Comparied to FL, SL provides better model privacy because the ML model is split between the clients and the server, which is useful for two reasons [35]. It offers model privacy since the users, and the server has no access to each other's model. Also, the processing workload at the client side can be significantly reduced by assigning computation of only a small part of the network to the clients considering the clients' capacity, which makes this method more suitable for resource constrained devices. In terms of speed, because of the sequential nature of ML model training across the in SL, it is significantly slower than FL.

However, there remain questions regarding whether privacy leakage exists stemming from the cut layer activation information sent by the client to the server during the training process as the activation information may leak a piece of information about the training data. In [36], the authors showed that it is possible to reconstruct the raw data from the activation values in the intermediate split layer, which are passed to the server. In their threat model, the server is honest-but-curious and tries to reconstruct the raw data from the activated vector of the cut layer.

E. COLLABORATIVE ML CHALLENGES

A brief comparison of different key aspects of collaborative ML approaches is given in Table 2. Most of the collaborative

TABLE 2. Comparison of Collaborative ML Methods.

	CL	FL	SL	DecL
# of Communication Rounds	One Round	Multiple	Multiple	Multiple
Shared Data	Training Data	Model Weights	Parameters in Cut Layer	Model Weights
Computation Overhead on Clients	No	Local Training	Partial Local Training	Local Training
Privacy Awareness	No	Yes	Yes	Yes

ML applications try to solve an optimization problem. Challenges for distributed optimization problems are inherited in collaborative ML as well. However, there are some differentiating properties as pointed out in [22] that summarize these aspects in FL context. hlThese apply to all collaborative ML approaches, as given below:

- Non-IID (identically and independently distributed) data: The training data on the clients, especially for cross-device settings, may not represent the overall distribution. IID sampling of the training data ensures the unbiased estimate of the full gradient in SGD. In neural networks, training with asymmetric non-IID data, decreases the accuracy significantly by up to 55%, where a single class of data is trained on each client device [37]. It is not realistic to assume that on each client, the local data is always IID.
- Unbalanced data size: The local training data size may vary for each client, e.g., the usage of the mobile service or application by some users may be much heavier than the others, or the data produced by one institution may dominate others.
- Communication constraint: Unavailability of client devices, dropouts, and synchronization latency resulted from network connectivity and power source issues, and computing constraints impact the algorithm performance. Increasing communication efficiency is a challenging task, and it may be needed to redesign the algorithms, e.g., reducing the communication cost of transferring large weight matrices of deep networks [38]. For wireless communications, communication media brings noise, which affects the convergence of the algorithm.
- Privacy: Comparing with centralized training, collaborative learning approaches have privacy benefits by allowing the model to be learned cooperatively without sacrificing data privacy. The exchanged updates contain minimum information about the raw data, and the aggregation algorithm does not need to know the source of the updates. However, if the parameters and architecture are not protected, an adversary may reconstruct the raw data. Furthermore, malicious users may cause more security problems, as discussed in section V. As a result, further protection of parameters and studying the trade-off between privacy level and the system performance are still needed [17].

For collaborative ML methods, in term of communication efficiency, SL is more efficient when the number of clients is increased and is highly scalable with respect to the number of model parameters. On the other hand, FL is more communication efficient when the number of data samples are increased but the number of clients and model size are small [39]. SL outperforms FL in terms of accuracy and requires lower computation resources per client. For setup with a large number of clients, SL requires lower computation bandwidth per client in comparison to FL [34]. Decentralized learning does not need a central authority to be trusted as federated learning does. Additionally, DecL is more resilient than FL because there is no single point of failure as in FL. However, DecL is generally slower to converge compared to FL [40].

V. SECURITY AND PRIVACY CONSIDERATIONS FOR COLLABORATIVE ML

In this section, we first give a brief introduction to our threat model in a collaborative ML setting and focus on each collaborative ML model. Then, we cover security attacks in ML to make this work more comprehensive. Finally, we explain existing privacy attacks in machine learning.

A. THREAT MODEL

This section examines potential threats in collaborative machine learning, which enable us to understand the attack model and construct mechanisms to defend the ML process against attacks from a privacy perspective [41]. For collaborative ML, we can consider training dataset, the model parameters, and hyper-parameters as assets which are sensitive and at risk of attacks. The data owners, the model owner, the model consumers, and the adversary are those actors which are specified in our threat model. The data owners own the data and are not willing to share the data due to security reasons or privacy considerations. The model owner who does not necessarily own the data, creates an ML model from the training data using data mining and machine learning techniques. The model owner is unwilling to share its model with other parties to prevent model inversion attacks and does not want to create a poisoned model because of the attacks from malicious data owners. The model consumers are those who use the service provided by the model owner through some programming or user interface. The adversary, as a usual consumer, may access the interfaces and access all communication between the parties. Additionally, the adversary may have a priori knowledge of data or models.

Adversarial knowledge is a key factor in determining variable attack surfaces against ML models. The adversary might have limited, partial, or full knowledge of model architecture, hyper-parameters, or training setup. From the dataset point of view, in majority of the works, it is assumed that the adversary has only some knowledge about the data distribution but not the training data samples. Attacks can be classified as black-box, white-box, and partial white-box with respect to the knowledge of the adversary [42], [43]. In black-box attacks, the adversary does not know about the model parameters, architecture, or training data [44], [45]. MLaaS, is an example of a black-box system where cloud hosts a pre-trained model, and users can query the model and learn a prediction vector or a class label. White box attacks are type of attacks that adversary has full knowledge about the model, including ML algorithm, ML model parameters, and architecture during training. Partial white-box attacks, which are also referred to as gray-box attacks, are those attacks where the assumptions are stronger than black-box setting. The adversary cannot access to the model parameters as in the white-box attacks.

We consider two main types of possible adversarial behavior known as honest-but-curious and malicious [46]. The honest-but-curious adversaries are the adversaries who follow the protocol steps but want to learn more information about private inputs of other parties from the messages received in the execution of the protocol. They try to infer information about the data during or after the training. The malicious adversaries can be defined as the adversaries who may not follow the protocol steps to recover more information about private inputs of other parties or to prevent the execution of the ML model creation or lead to wrong model creation. For example, they can inject non-legitimate messages in the protocol transcript, may not execute a protocol step, and may send a message more than once.

There are two kinds of attacks against machine learning, considering the intent of the attacks. One of them is security attacks which do not intend to gather information about sensitive data but try to influence the model's thinking to incorrect output prediction such as poisoning and evasion attacks. The other is privacy attacks, where the goal of the adversary is to gain more information about the sensitive data, such as the training data and the ML model, rather than gather the information that can be retrieved from the output. We explain them in the following sections.

B. THREAT MODELS FOR COLLABORATIVE ML

In the following, we discuss privacy attacks' design for each collaborative ML model from adversarial behavior view. A malicious and honest-but-curious adversary are represented by black and white evil respectively in the figures. The adversary can reside at the aggregator server, or client side.

Federated learning mostly employs federated averaging or Synchronous SGD [47] algorithms. In both algorithms, the global model parameters can be accessed by each client, and model parameters or loss gradients can be obtained by the server. As illustrated in Figure 5, in federated learning, a malicious adversary represented by the black evil could be either a client or the server. A malicious client can observe the global parameter updates and creates his own adversarial parameter updates and gain knowledge about aggregated training data of all participants [48]. Alternatively, the malicious server can inspect clients' update and tamper training process by modifying each client's view on the global model and extract more information about training data.

However, the honest-but-curious (white evil) server only observes the updates and wants to gain information about clients' local data. In addition, an honest-but-curious client can only observe global updates and launch an attack to gain knowledge about other clients' local data.

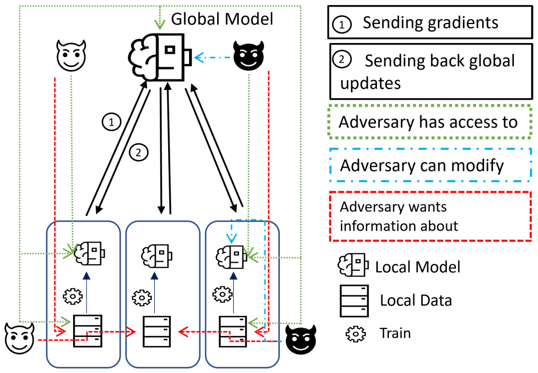

FIGURE 5. Threat model against FL.

In Figure 6, where we describe decentralized learning attack model, there is no aggregator server, and all nodes are going to send updates to each other. A malicious node observes updates from all other participants and can modify its own parameters to tamper with the training process and gain knowledge about other nodes' data. However, an honest-but-curious node can only observe the updates and the global model to launch the attack to extract information about training data.

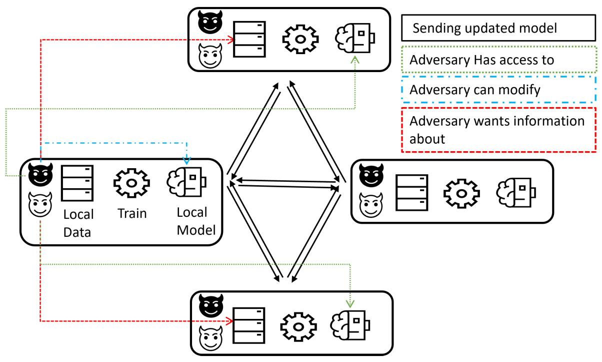

FIGURE 6. Threat model against decentralized learning.

In Figure 7, the attack model against split learning is depicted. The honest-but-curious server follows the operations as specified, and it wants to gain information about the raw data stored on the client. The server has access to the activated vector of the cut layer sent by the participants during the forward propagation. It aims to reconstruct the raw data of the clients' data. Alternatively, an honest-but-curious node has access to the gradients sent by the server during the backward propagation and wants to gain information about the data of other clients. However, the malicious server can use each node's activated vector of the cut layer and modify gradients during the backward propagation to tamper the training process and extract more information about training data.

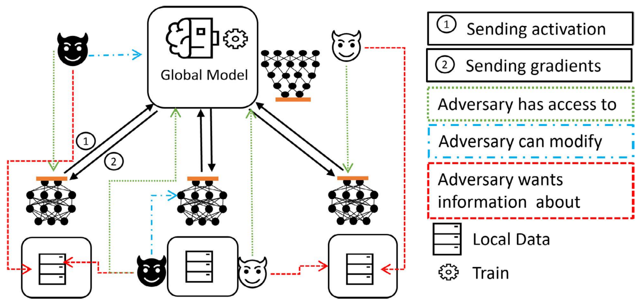

FIGURE 7. Threat model against split learning.

C. SECURITY ATTACKS

Although we focus on privacy attacks in this work, we present security attacks briefly for the sake of completeness. Security attacks, also called as adversarial attacks, attempt to create a misbehaving model [17], such as poisoning and evasion attacks, [20], [41].

In Figure 8 we also provide a classification for security and privacy attacks based on their occurrence in the training or inference phase.

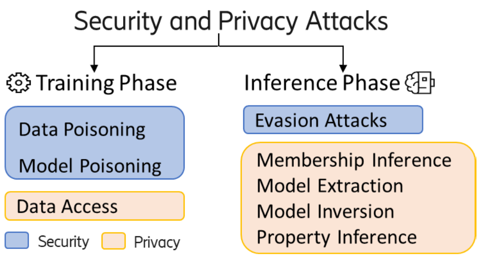

FIGURE 8. Classification of Security and Privacy attacks.

1) POISONING ATTACKS

Poisoning attacks mostly occur in the training phase. In poisoning attacks, the data can be altered directly by data injection or manipulation; logic corruption can also alter the model directly [41]. The adversary can modify the training data by adding adversarial entries to the original training data aiming to reduce the accuracy of the model, which changes the underlying data distribution without changing the features or labels of the data. The adversary may also want to modify the output labels or input features of training data or also may want to alter the ML model directly by tampering with the ML algorithm process. Poisoning attacks enable adversaries to insert backdoors or trojans to the model either at training time or after initial model training [49]. For example, Gu *et al.* [50] inserted stop sign images with a special sticker (the backdoor trigger) into the training set and labeled them as speed limit signs. In this way, a backdoor in a street sign classifier is generated where the common street signs are classified properly, but the stop sign possessing the backdoor trigger are incorrectly classified as speed limit signs. Thus, simply by placing a sticker on any stop sign, the adversary can trick the model to classify it as a speed limit causing potential accidents in self-driving cars.

The invisibility of model updates generated by each client leads FL to be vulnerable to model-poisoning attacks. In order to add backdoors to the joint model, a malicious client can use model replacement. The adversary can act as a single client or by colluding with multiple clients to modify a classifier to assign desired labels [51]. In FL, user-level differential privacy can be used to defend against targeted poisoning attacks. In [52], the authors implement the sybil attack, a model poisoning attack on differential privacy based federated learning and explore some protection mechanisms. The adversary arranges manipulation of model updates by creating several fake clients or colluding compromised clients.

2) EVASION ATTACKS

These attacks occur during the inference/testing phase. The adversary aims to perturb the input samples at inference/test time to ML classifier to cause a misclassification. For example, the adversary can change some pixels in the image of the 'stop' sign, which causes it to be predicted as a 'Speed Limit' sign by the classification model. In [53], the authors proposed data transformations, including dimensional reduction as a defense mechanism against evasion attacks. They demonstrate that the adversarial success rates are reduced at a fixed budget but are not completely solved.

D. PRIVACY ATTACKS

Privacy concerns in machine learning may arise in many cases such as sharing a public dataset, participating in a training procedure using sensitive data to generate a model, sharing the learned model publicly, and sharing query results with the end user. In all such cases, the privacy of an individual's data or a service provider's model is at risk. In the following, we explain such attacks which can arise both at training or testing phase of machine learning.

1) DATA ACCESS ATTACKS

These kinds of attacks can be executed by one or more collaborating parties who want to learn sensitive information about other parties' data. For example, in the federated learning scenario, the server sends the updated model to the data owners in each iteration. The difference between two consecutive models sent by the server can be used by semi-honest or malicious data owners to recover some information about the private inputs of other data owners. Alternatively, the server can be honest-but-curious and tries to learn some extra information about the private inputs, for example, by linking the model updates of each client. There are solutions to mitigate such attacks, like secure aggregation and making FL server oblivious to hide the data owners' identities [54].

2) MEMBERSHIP INFERENCE ATTACKS

This attack was first introduced by Shokri *et al.* [55], and it is one of the most popular categories of attacks. This is an oracle attack where the adversary can infer whether a specific record belongs to the training dataset by utilizing the difference between the model's confidence on data records that were or were not seen during training. Membership inference attacks against ML can be quantified by investigating how much information that machine learning models disclose about the data records used in the training phase [55]. That is, with black-box access to a model and data, the attack specifies whether a record existed in the training dataset of the model. The target model can be queried with a data record, and the prediction of the model on that record is obtained by the adversary. The prediction is represented by a vector where each element represents the probability that the record belongs to a certain class. In [56], the authors investigate membership inference attacks under a black-box access scenario in which the trained model is private, and an adversary may only query the prediction API and receive the prediction output. They show that it is a complex strategic process to generate a membership inference attack model. Also, to understand why certain models and dataset are more vulnerable, and when and how these kinds of attacks work, they study these kinds of attacks across different target model types and different type of training dataset. Using attribute inference (guessing type of data), valuable information about training data can be extracted.

Nasr *et al.* [48] showed that the membership inference attack is more effective than the black-box one when data from the training dataset is accessed by the adversary. In the honest-but-curious scenario, in order to differentiate between members and non-members, the model parameters and gradients are used as an input to train another model. In the malicious case, instead of gradient decent, the adversary can perform gradient ascent by altering the gradient updates for the data whose membership is questionable. If the data is used by some other participants for training, the gradient of the loss will be reduced significantly by other participants' local SGD, which as a result affects the updated model and lets the adversary to extract membership information as presented in the attack in [19]. The attack accuracy from a malicious local participant reaches 76.3%. The effect of the number of participants on attack accuracy is adverse, i.e., after five or more participants, the accuracy notably drops. However, the attack accuracy from a malicious server, which is in a more desirable position, reaches 92.1%.

3) MODEL EXTRACTION ATTACKS

These attacks, also known as exploratory attacks, are oracle attacks where the goal is to obtain parameters or structure of the model by inspecting the model's predictions, including the probabilities returned from each class. An adversary with access to prediction API or model outputs tries to rebuild a surrogate model that approximately matches the target model. This attack can be implemented by querying the prediction API and learning the predictions for the input feature vectors. When there is no constraint in the number of queries and the queries themselves, the adversary can construct a model similar to the target model by querying many times and using the inference results as training inputs. One of the trivial solutions to prevent such attacks is to limit the number of queries from users. The most obvious countermeasures for ML services are to remove the confidence values and only output the class labels [6]. Embedding a watermark can be another approach for IP (Intellectual property) protection of ML models to determine that the model is stolen. Watermarking is regarded as selecting a set of inputs (i.e., a trigger set) that are labeled randomly and can be used by a legitimate model owner along with normal training data to generate a watermarked model. To demonstrate ownership of the model, the surrogate model is queried with a trigger set. If incorrect labels are matched with enough predictions, it can be concluded that the model has been stolen. In this way, a legitimate model owner can detect misuse of their models [57]. In addition, in [58] the authors propose an active defense that perturbs prediction to poison adversary's training objectives. They claimed that the accuracy of the adversary could be decreased by up to 65%, where the defender accuracy is not affected significantly.

4) MODEL INVERSION ATTACKS

A model inversion attack is another oracle attack type that uses a priori information about the model and auxiliary data to explore training data or other sensitive data. The inferred information enables the adversary to reconstruct the data sample used to train the model, which may violate the privacy of an individual whose personal information is included in the data [41].

5) PROPERTY INFERENCE ATTACKS

Property inference is the ability to infer properties other than those explicitly encoded as features where the model producer did not intend to share. Inferring the fraction of the data that comes from a certain class, for example in a patient dataset the aim of attacks is to infer the fraction of men and women when such information was not an encoded attribute. The information was learned unintentionally from the model, which is not related to the training task [19]. In collaborative settings, one of the clients can be maliciously trying to infer the uncorrelated features of the dataset, e.g., if the model is for gender classifier, adversary tries to infer the facial id of the picture. The attack can be performed passively or actively. In the passive setting, the adversary saves the snapshots of the joint model at different rounds reflecting aggregated gradients. Then, the adversary calculates the difference between these gradients and tries to infer information based on the assumption that the gradient updates can leak the features of the input data learned by the model to predict the output. In the active setting, the adversary attaches an extra classifier for inference e.g., for facial id inference and crafts the gradient updates so that it can infer extra information. The attack, which is inferring properties from observation, is also a learning problem.

VI. PETs FOR COLLABORATIVE PPML

In this section, we give background information about the main privacy enhancing technologies that can be utilized to prevent the privacy attacks under the threat models presented in the previous section. We focus on the following PETs: differential privacy, homomorphic encryption, secure multi-party computation, and confidential computing techniques. We provide a state of the art for the PETs and how they are used in collaborative ML settings. Note that secret sharing can also be included in the list but throughout the paper we consider the secret sharing primitive in the secure multi-party computation where this primitive is widely used in this computation paradigm. In the differential privacy technique, noise is added to the data before sending it to another party to prevent information leakage. The secure multi-party computation and homomorphic encryption are two different approaches to allow computation on encrypted data. In the secure multi-party setting, the involved parties collaboratively compute a function of the private inputs of the parties. Homomorphic encryption schemes allow computation on ciphertext without needing decryption of them. Finally, the confidential computing approaches enable trusted executions based on the hardware guarantees on isolated and protected memory regions.

A. DIFFERENTIAL PRIVACY

Differential privacy (DP) is a data anonymization technique that enables privacy measurement by bringing some mathematical definitions. It is widely used with the objective of protecting the privacy of the individuals whose information is in a dataset in the context of statistical and machine learning analysis [59]. In this sense, it is a new approach for protecting privacy in a more quantifiable way than the literary norms that usually appear in many laws, policies, and practices [60]. There is increased attention to DP solutions as they can prevent several types of security and privacy attacks such as poisoning, model extraction, model inference, and membership inference.

Definition 1: DP is defined formally as follows [61]. A randomized algorithm M is ϵ-differentially private if for any subset of the output S in the range of M, and for all dataset D_1 and D_2 differing on at most one record:

$$Prob[M(D_1) \in S] \leq exp(\epsilon)Prob[M(D_2) \in S] + \delta \quad (1)$$

The given formulation is called (ϵ, δ)-differential privacy where δ is the relaxation parameter. If δ is neglected, then ϵ-differential privacy provides stronger privacy guarantees. $\epsilon$is the control parameter for privacy level, denoting privacy budget. There are other approaches like Rényi differential privacy (RDP), that is, an algorithm is (α, ϵ)-RDP if the Rényi divergence of order α between any two adjacent databases is no more than ϵ. RDP can be preferred for its simple privacy budget accounting. A more detailed mathematical foundation can be found in [62].

In FL context, D_1 and D_2 dataset relate to client training sets that are adjacent if D_2 can be formed from D_1 by inserting or removing all of the training samples associated with a single client. This approach is called user (or client) level privacy [63], where standalone ML DP applications are called example level privacy [64], [65].

DP applications can be classified as Central DP and Local DP depending on the FL trust model. In the FL setting, given in Figure 2, which uses central trust model, the FL server collects the client updates in cleartext. Central DP applications employ the same trust model but add noise on the server. In this way, the server sends the privacy protected model parameters. After receiving from the server, clients perform local training, clip their updates to bound their contribution, and then send the clipped updates to the server. Server aggregates the updates and again adds noise proportional to sensitivity. In each round of FL this process is repeated.

Three mechanisms can be used for noise sampling in DP, Laplace [66], Gaussian, and the exponential [61] mechanism. Gaussian is the most widely used mechanism in Central DP and defined as:

For a query function f:D$\longrightarrow$R a randomized algorithm M satisfies (ϵ, δ)-DP if

$$M(D) = f(D) + N(0, \sigma^2) \quad (2)$$

where N indicates the noise from Gaussian distribution with the standard deviation of σ calculated from sensitivity of the f.

Central DP requires clients to trust the server as the server controls the aggregation and DP mechanism. If this is not the case, Central DP can be improved from the privacy point of view by adding noise on client side so that clients don't have to trust the server. This model is called Local DP. This removes the trusted FL server assumption, on the other hand, reduces the utility [67], [68], [69].

Local DP allows the individual client to set different local privacy guarantees and was first formalized in [70]. However, it was also introduced in the database community, by Hsu *et al.* [71] with the name "amplification", and it gained attention with the work of Duchi *et al.* [72]. Local DP has been deployed effectively on end-user device application and real-world deployments gave rise to its popularity. Google's Rappor [73] framework uses Local DP to protect individual's browsing habits to identify popular destinations and settings. Apple, [74] uses Local DP to discover typing habits to improve keyboard auto-correction service and to identify energy and memory consumption of the browser. Microsoft [75] also uses Local DP to collect telemetry data to improve user experience.

Definition 2: The Local DP guarantee is formalized as follows [70]. For a given $\epsilon \in R+$, a randomized mechanism M satisfies ϵ-LDP if and only if for any pair of inputs x_1, x_2, and any output y, the probability ratio of outputting the same y should be bounded

$$\frac{Prob[M(x_1) = y]}{Prob[M(x_2) = y]} \leq e^{\epsilon} \tag{3}$$

In the literature, Local DP (LDP) in FL has gained attention to address the honest-but-curious aggregator threats after its introduction in [66]. Different LDP mechanisms have been suggested for matrix factorization [76], key-valued data [77], [78] and multidimensional data [79], [80], [81], [82], [83]. In [84], the authors proposed an LDP algorithm in vehicular communication domain to predict the traffic status using crowd-sourcing applications for numerical data represented as a single numeric attribute. They integrated the LDP algorithm to FL, called as LDP-FedSGD. Wang *et al.* addressed the problem of perturbing multidimensional numeric and categorical data in [79] with optimal worst-case error. They proposed two methods for single attribute numerical data, called Piecewise Mechanism and Hybrid Mechanism to improve previous work by Duchi *et al.* [80]. Then, they extended these methods to handle both numeric and categorical attributes for multidimensional data. Similar to [79], the study in [85], provided a new approach for LDP to handle high dimensional, continuous, and large-scale model parameter updates during FL while allowing the clients to customize their privacy budget.

In [86], the authors concentrated on the efficiency problems of LDP stemming from the high variance of the noises. This leads to more communication rounds between server and clients and more privacy costs to achieve desired results [85]. Their design provides splitting and shuffling to each clients' gradients just before sending to the server to mitigate the privacy degradation caused by multidimensional data and a high number of iterations.

[87] adapted robustness and privacy perspective to analyze how both CDP and LDP mitigate the backdoor, membership inference and property inference attacks with an experimental framework. Their results show that LDP is ineffective for property inference attacks while CDP can defend with a significant loss in utility. CDP and LDP protect against backdoor [88] and membership inference attacks [55].

Recent works try to balance Central DP and Local DP to have higher utility without too much privacy loss using distributed differential privacy (DDP) approach. In DDP, FL is not necessarily used, and other security mechanisms support the Local DP mechanism as in the secure aggregation protocol. The clients protect their updates locally via Local DP, and secure aggregation ensures that the FL server does not reveal the intermediate parameters [89], [90], [91]. In [89], the authors tailored the setting to fit the FL by taking the FL client drop-out challenge into account. Another approach in DDP is to use Shuffling [92], [93], [94] and Mixnets [95]. These approaches try to hide the link between the data and the client. [94]m presented Encode, Shuffle, Analyze (ESA) architecture with the idea "hide in the crowd". Basically, clients first perform Local DP, then the output is securely shuffled and sent to the server. In shuffling methods, privacy can be adjusted with the use of moderate ϵ to enable the protocol with far smaller error than using only Local DP, while not solely relying on the trusted server model.

The preference and trade-off between CDP and LDP come from the trust model of the deployments. CDP cannot provide privacy protection in cases of the malicious server model. Although LDP does protect the clients from the malicious server, it reduces the accuracy of the model. Moreover, the malicious colliding client model is not taken into account in the DP itself. Bringing other privacy enhancing techniques with a hybrid solution would be an alternative way. Without sacrificing accuracy, it is still possible to protect from the malicious server model in the solution using SMPC and HE. On the other hand, these approaches come with a price, inducing additional communication, and computation costs.

DP allows controlling and tracking the privacy with the moments accountant method [96] so that the defined privacy budget given with the via (ϵ, δ) parameters is not exceeded. In collaborative learning, the iterative nature of the training algorithm should also be reflected in privacy accounting. Privacy accounting for multiple iterations can be done using the composability feature of DP to compute and accumulate the privacy cost at each round of training.

B. HOMOMORPHIC ENCRYPTION

Homomorphic encryption (HE) is a type of encryption that enables computation on ciphertexts without access to the secret key or need for decryption; when the encrypted result is decrypted, it matches the result of operations as if performed on plain text. Unlike other encryption algorithms in use today, lattice-based HE [97] algorithms are claimed to be safe against quantum computer attacks. A public key is used to encrypt the data, and the algebraic structure in lattice-based HE systems is utilized to allow functions to be performed directly on the encrypted data. After applying the functions to the encrypted data, the result can be accessed only by the party that owns the private key. HE includes different types of encryption schemes that can perform different classes of computations over encrypted data [98]. Commonly known homomorphic encryption types are partially homomorphic encryption (PHE), somewhat homomorphic encryption (SWHE), and fully Homomorphic encryption (FHE) [98]. The computations are represented as either Boolean or arithmetic circuits. PHE [99] supports an unlimited number of operations of only one type (i.e. addition or multiplication) on the ciphertext. Due to its relatively low computation and storage overhead, PHE-based algorithms are used in many practical applications, although they support only one kind of operation. PHE can be divided into two types: additively homomorphic encryption and multiplicatively homomorphic encryption. One example is Paillier cryptosystem [100], where only addition operations can be performed on encrypted data and for multiplying values under encryption, additional interaction with the party having access to the private key is needed. For multiplying values under encryption, additional interaction with the party having access to the private key is needed. The Paillier cryptosystem is widely used for keyword search on the remote encrypted data, privacy-preserving aggregation, etc. SWHE [101] supports all sorts of arithmetic and logic operations. The most important drawback of SWHE system is that the number of homomorphic operations is limited. Another limitation of SWHE is that application of all operations to all types of data is not possible at the same time. SWHE is suitable for a variety of real time applications such as financial, medical, and recommender systems. Since SWHE supports a limited number of operations, it will be much faster than fully homomorphic schemes. FHE [102] scheme supports arbitrary operations with an unlimited number of times over encrypted data. The main problem of FHE is costly operations, including bootstrapping step used to reduce the noises in the ciphertext. If the applications are time-critical and the message size is big, then using FHE can be impractical. Due to computational overhead, FHE is less efficient than PHE and SWHE. As of today, for a particular application PHE and SWHE schemes are much more practical compared to FHE schemes.

Ensuring data privacy is a highly valuable advantage of HE. Without decrypting the data, HE allows multiple computations to be done on encrypted data. HE is especially useful for privacy-preserving computation over data, whose storage is outsourced to third parties. In particular, after having been homomorphically encrypted, data can be outsourced to a commercial cloud storage service and processed while it is encrypted. HE can be used in highly regulated industries such as health care to enable new services, where data barriers can be removed by inhibiting data sharing. Homomorphic encryption can also be used in the delegated computation of trained ML models. In this approach, one of the parties encrypts data and sends it to another party to process it and return the result. None of the parties can gain information about other parties' training data other than what can be inferred.

Although HE provides stronger privacy protection, there are three main weaknesses of HE schemes [103]. 1) High computational cost: the efficiency of the system is decreased by requiring several modular exponent arithmetic for designing the secure computation protocol for most of PHE and SWHE schemes, also requiring bootstrapping technique for reducing the noises from the ciphertext in case of FHE. 2) Large storage overhead: comparing with the original plaintext, the storage cost for ciphertexts will be expanded many times (e.g. hundreds or even thousand times) for most of FHE scheme and some PHE and SWHE schemes. 3) requiring trusted authority (TA): TA is responsible to generate and distribute public/private keys for all the parties.

In the context of collaborative learning, in order to learn models privately using homomorphic encryption, parties first need to distribute keys. Below we first explain about key distribution in homomorphic encryption procedure between parties in collaborative learning.

In one scheme, a typical cross-silo FL system is mentioned [104]. Each client has an HE module, and there is an honest-but-curious aggregator server that coordinates the clients and aggregates the encrypted gradients. A cryptographic protocol such as SSL/TLS protocol is used to secure the communication between the clients and the aggregator server; thus, the transferred messages cannot be learned. One client is selected randomly by the aggregator as the leader to generate an HE key pair. The selected random client synchronizes the keys to all the other clients; also generates the ML model initially and sends the weights to other clients. Whenever the clients receive the key-pair and the initial model, they start training and computing the local gradient updates. Clients encrypt the updates using the public key and send the results to the server. The server performs homomorphic computation on all received updates (e.g., add them up) and sends out the results to all clients. The aggregated gradients are decrypted by the clients and the local models are being updated. The assumptions behind this scheme may be considered prohibitively strong; it is assumed that the server will not collude with any client. If the server chooses a client intentionally as a leader, it can learn the HE key pair and consequently learns gradients sent by other clients. In addition, the leader client has to communicate with all other clients to distribute the key-pair which can increase the communication cost.

In another scheme, a Multiparty Homomorphic Encryption (MHE) [105] uses HE scheme to encrypt and exchange the input data between multiple parties. Using some secret-sharing scheme, the secret key is distributed securely among the participants to preserve the privacy of inputs. The participants need to collaborate with each other for the decryption process according to the access structure of the used secret sharing scheme. In this scheme, the clients collaboratively generate a private key. Even if the server colludes with one client, it could not get information about gradients of other clients.

The multi-key HE scheme is an important class of Multiparty HE which is firstly proposed by López-Alt *et al.* [106]. In this scheme, computations can be made on ciphertexts which are encrypted under different and independent keys without a joint key setup. The decryption of the ciphertext can be done jointly by all users who are involved in the computation.

Now, we give an overview of existing solutions in the literature, which enhance privacy in collaborative ML using homomorphic encryption.

Mittal *et al.* [107] proposed a secure k-means data mining approach where the data is assumed to be distributed among different hosts (i.e., horizontally partitioned and stored). Their approach is to mine data securely using k-means algorithm from the cloud even when adversaries exist. The privacy of data at each host must be preserved, and no intermediate values can be leaked to the adversary. Thus, each host only knows about its inputs and the final outputs, but not the intermediate values. They use Pallier cryptosystem for the privacy of the data of each host and also the intermediate results.

In [28], the authors aim to perform a federated logistic regression in the feature space. They proposed an end-to-end three-party solution for situation where data is vertically partitioned between two organizations. A linear model will be learned collaboratively in a federated setting. They propose a secure protocol where additively homomorphic encryption and privacy-preserving entity resolution is employed by the coordinator (i.e., a third party), assuming the exchange of raw data is not possible. The values in each organization's data are protected from another organization using additively homomorphic encryption. Protection from the coordinator (who has the private key) is provided by sending only calculated values (such as gradient) that are not considered private. In [108], using homomorphic encryption, the authors proposed a secure system for protecting both the training and predicting data in logistic regression. Data from distributed devices (as in the Internet of Things) can be securely handled by their design; a central server might be needed to receive, store, and process the data. Using homomorphic encryption, plain data is never disclosed to the server.

In [109], the authors presented a privacy-preserving deep learning system, in which local data of participants are not revealed to a central server, and learning participants perform neural network-based deep learning over a combined dataset of all. Using additively homomorphic encryption, they protect the gradients over the honest-but-curious cloud server. The additive homomorphic property makes computation possible over gradients encrypted and stored on the cloud server. There is a trade-off between protecting the gradients against the cloud server and increased communication between learning participants and the cloud server. Their provided system does not reduce the accuracy of deep learning. In [110], the authors proposed the first federated transfer learning framework for wearable healthcare. They used federated learning and HE to aggregate the data from different users/organizations to build machine learning models while the users' privacy is preserved. Using a public dataset, the cloud model is constructed on the server and distributed to all users/organizations. Each user model is trained based on the shared cloud model and its own data. The user models are uploaded to the server for aggregation using HE, and the new cloud model is trained. Using cloud model and local data, each user can train personalized models. All parameter-sharing processes are made through HE. In [28], the aim is to learn a linear model collaboratively in a federated setting, where data is vertically partitioned between data providers. The shared model is trained using locally computed updates. They propose a secure protocol managed by a coordinator that implements privacy-preserving entity resolution and an additively homomorphic encryption scheme. The third party holds a private key and receives only the encrypted aggregated model updates from participants which are not considered private in their setting. They provided a formal study on the impact of entity resolution errors on learning since identifying corresponding entities in a vertically partitioned dataset is challenging.

In the cross-silo FL framework, to ensure that clients' updates are not revealed during aggregation, they are allowed to mask their local gradient updates using additive HE. However, the computation and communication cost of HE operation is extremely high. In [104], the authors proposed a BatchCrypt, a simple batch encryption technique, to solve the communication and computation bottlenecks caused by HE. Each client represents its gradient value with low-bit integer using quantization. Then the batch of quantized values is encoded into a long integer, and then batch encryption is performed, which decreases the overhead of encryption and total volume of ciphertext. The authors validated their techniques with experimental results.

In [111] and [112], the authors proposed privacy preserving multi-party machine learning based on federated learning and homomorphic encryption where each node has a different HE private key in the same FL-based system. In [113], the authors proposed a privacy-preserving FL approach which use a momentum gradient decent optimization algorithm (MGD) to accelerate the model convergence rate during the training process. To preserve the local privacy information of each agent, a fully homomorphic encryption is adopted to encrypt gradient parameters.

C. SECURE MULTIPARTY COMPUTATION

Secure multi-party computation (SMPC) is another paradigm for computation on encrypted data in addition to the homomorphic encryption technique. In the SMPC setting, there are parties with their sensitive inputs, and they want to compute a joint function using their inputs but don't want to reveal their private inputs to each other. In other words, at the end of the protocol, the parties learn nothing beyond what is revealed by the output itself. The term ''Secure two-party computation'' is used for the special case that the number of parties is just two. Note that in this computation paradigm, there is no need for a trusted server/party.

Collaborative ML training can be considered as a function that receives sensitive data of data owners as input and outputs the collaborated machine learning model itself to the data owners or to a server. The output of the SMPC can be inference results for given inputs instead of the model itself.

In literature, there are two generic secure multi-party protocols that can execute any function securely. One of them is Yao's garbled circuits introduced by Andrew C. Yao in 1986 [114], which works only for a two-party case. In that solution, the sending party creates a circuit for the function needed to be computed, randomly selects two symmetric keys for each wire in the circuit for two possible values '0' and '1', then sends the truth table for each gate in the circuit in a random order, with the corresponding keys for its inputs. The receiver party gets the corresponding keys for its input bits obliviously by running oblivious transfer (OT) protocol with the sender party and then evaluates the circuit by decrypting the garbled truth table of each gate.

The second generic secure multi-party computation protocol is GMW protocol proposed by O. Goldreich, S. Micali, and A. Wigderson [115]. Similar to Yao's garbled circuit, the function needs to be computed is represented as a circuit of XOR and AND gates. Each party divides their inputs into shares and sends the shares to each other. For XOR gates, it is free to compute the output because XOR of shares gives the shares of the XOR result of inputs. For the computation of AND gate output, the parties have to run the oblivious transfer protocol. The OT protocol, first proposed by Michael O. Rabin in 1981 [116], is a two-party protocol that is widely used in secure multi-party computation, as two examples are given above. In the OT protocol setting, a sending party who holds two messages sends only one of the messages depending on the choice of the receiver party, without knowing any piece of information about the choice. OT protocols use public key cryptographic algorithms that need more computation time and resources when compared with symmetric key algorithms. However, with the development of OT extension algorithms [117], the number of public key operations is reduced to constant numbers such as 128 to execute many OT operations. With this development, the computation cost of secure two-party and multi-party computation protocols is decreased dramatically. However, to execute OT and other necessary operations, more than two communication rounds and transfer of messages are needed, which increases the communication cost of secure multi-party computation protocols. Although there is a recent improvement in the number of rounds of OT extension protocol, which reduces the number of rounds from three to two, the setting requires some offline operations between parties [118].

While Yao's garbled circuit protocol and GMW protocol can compute any function, sometimes it is preferred to design function-specific secure multi-party computation protocols considering computation and communication complexities of the generic protocols. Secure equality testing, comparison, and private set intersection protocols can be given as examples for such specific functions. In secure equality testing protocols, two parties holding private inputs can learn equality results without revealing their sensitive data to each other. Secure comparison protocol is similar to the secure equality testing protocol except that the output is the comparison result. In the secure private set intersection protocols, the parties holding a set of items learn the intersection of the sets without revealing the items that are not in the intersection to each other. These functionalities (secure equality testing, comparison, and private set intersection) are commonly used functionalities in protocols, and because of that, many specific constructions have been proposed for these specific protocols.

Collaborative ML setting can be regarded as a specific SMPC setting where the function to be computed in the SMPC is the ML model training or ML model inference where the inputs are the training data used in the model training or the query for the inference, respectively. At first glance, it can be seen that SMPC solutions can be costly because of heavy cryptographic operations and the need for communication between data sources as pointed out in [11]. It can also be concluded that SMPC techniques mitigate only the privacy attacks as stated in [21] and [11] for FL. Nevertheless, there are some solutions and advances to mitigate these drawbacks of SMPC and allow the utilization of this privacy enhancing technology. For example, to not require communication between data owners, a trusted authority, who distributes necessary keys to the participants and then leaves the protocol, can be used. Such a solution for secure aggregation was proposed in [119]. Also, with the deployment of oblivious extension transfer techniques [117], the computation cost of cryptographic operations can be decreased. Regarding the other drawback pointed out in [21] and [11], it can be argued that in addition to mitigation of privacy attacks, SMPC techniques can support mitigation of security attacks by allowing execution of security attack mitigation steps without leaking any sensitive data. In this setting, the function to be computed in SMPC does not only include the ML model training and execution but also includes the security attack prevention steps. Such usage of SMPC is introduced in [120], where poisoning attacks are prevented without any sensitive data leakage. Some other solutions that use SMPC to protect the privacy and preventing backdoor attacks in federated learning are introduced in [121], [122], and [123].

Since collaborative ML is a function, the privacy of the parties' input can be protected by using generic SMPC solutions: Yao's Garbled Circuits or GMW. However, the realization of privacy preserving collaborative ML by using these techniques is not so practical because of heavy computation and communication costs. Because of that, custom SMPC protocols have been offered to enhance privacy in collaborative ML. Instead of the construction of the SMPC protocol for the whole training algorithm, some parts of the training algorithm can be implemented in the SMPC protocol. For example, in the federated learning scenario, solving the secure aggregation of weight updates can be enough to prevent sensitive data leakage. Commonly used primitives in construction of SMPC protocols are secure aggregation protocols [31], [124], secret sharing schemes, and usage of generic protocols (GMW and Yao's garbled circuits). In the secure aggregation setting, the server (called as aggregator) aggregates the sensitive data of the parties without learning any sensitive information about the data. Below we give a short overview of existing solutions in the literature.

For the federated learning method, [31] proposed a solution to prevent the server from learning the weight updates by obscuring the aggregation from the server. Another work related to federated learning setting proposed a method to privately aggregate the model parameter updates and detect malicious update values using secret sharing scheme [120]. In that solution, two non-colluding servers are required. Similar to that work, many of the solutions in literature make use of secret sharing schemes and usage of non-colluding servers such as [125], [126]. With that approach, [125] proposed new protocols for privacy preserving ML methods such as linear regression, logistic regression, and neural networks. In that setting, there are two non-colluding servers which are honest-but-curious and data owners distribute their data among the servers where none of the servers can learn any piece of information about the sensitive data without making any collaboration. The servers train the models using secure two-party computation. While most of the works in literature compute linear operations in a privacy-enhanced way such as secure aggregation, in that work MPC-friendly alternatives for non-linear functions such as sigmoid and softmax operations were introduced. Another work that uses secret sharing is [127], which was presented at ESORICS 2020. Yet another privacy solution for neural networks in addition to [125] can be found in [128]. [129] proposes privacy solutions for deep learning using Yao's garbled circuits. Also, some proposals combine SMPC and HE, such as the one introduced in [130]. That work proposes a privacy preserving prediction solution. Note that most of the solutions have been proposed under the semi-honest (honest-but-curious) adversary model. To make the protocols secure against malicious adversaries, some techniques such as zero knowledge proofs need to be used, which makes the protocols impractical as stated in [131]. Also, the solutions in literature can be categorized considering the number of parties involved in the SMPC protocol. For example, the proposals in [126], [132], and [133] use two-party SMPC protocol while [134] includes three-party communication. For the linear regression model learning, [135] proposes SMPC protocol using Yao's garbled circuits. Here the training data set is vertically distributed among parties.

D. CONFIDENTIAL COMPUTING

Confidential computing (CC) is a secure computation model that assures users that their environment is secure when they run applications on the virtualized environments benefiting from hardware security offerings. It provides hardware enforced isolation with memory encryption via enclave technologies and uses advances of trusted execution environments (TEE) to build enclaves. Enclaves are hardware-isolated and protected memory regions of the code and data that run on reserved part of the physical memory, so they cannot be accessed directly from RAM. The technology is provided mostly by hardware vendors like Intel, ARM, and AMD and can be named differently.

The trust model for secure enclave technologies ensures protection for malicious insider actors like malicious hypervisor on virtualized cloud environments. On the other hand, side channel attacks [136] to the processors that provide the security functionality are still possible. Although vendors release some countermeasures or patches, they do not guarantee that the processors are safe from side channel attacks. Mitigating the risk is left to the solution provider. Hence, the security functionalities and threat model might be limiting the intended security design, and the system is protected as much as the security of the vendor's solutions. As for the other outsourced services, integrity, confidentiality and privacy is a growing need for ML services that use remote or shared environment, no matter there are different actors involved in or not. The confidential computing paradigm enabled by TEE is a natural pragmatic solution to this problem as it provides secure training and inference for ML by isolating sensitive computations from the untrusted stack. The use of confidential computing comes at a price; it requires additional capabilities on hardware. In the FL setting, TEE can be used for server-side operations or can be used in client devices. If the client-side is a massive deployment environment like IoT cases, the requirement that each IoT device has TEE would be costly. Although there are IoT specific solutions like ARM TrustZone, current implementations are mostly done on server-side. On the other hand, recent *sybil* attacks show that protection against malicious client devices is required to prevent sybil based poisoning attacks [137].

Most of the recent works exploiting TEE in ML focus on providing privacy on Machine Learning as a Service (MaaS) setting, which does not use FL topology necessarily. For the studies in [136], [137], [138], [139], [140], [141], [142], [143], [144], [145], and [146], the authors focused on how to protect the DNN model in MaaS setting using TEE. They try to solve the problem arising from the fact that executing model inference within the TEE is not practical due to hardware constraints. The code running inside the TEE (enclave) is bounded by a threshold, e.g., 128 MB for Intel SGX. If the threshold is exceeded, then data swapping occurs creating performance and security issues as the data must be decrypted and encrypted during swapping operations. To overcome this problem, the model can be split up, and GPU accelerators are used for the untrusted code. Since GPUs do not provide trusted execution, how to outsource to GPU and how to split the DNN are the most important issues, which most of the studies try to deal with. To provide some level of protection, in some of the works, a blinding operation is performed before outsourcing the computation from enclave to GPU. After GPU complete with its operations and send back to the enclave in blinded form, unblinding is performed within the enclave [140], [143], [144], [145].

Depending on the trust model, there can be two scenarios where TEE usage can be leveraged in FL cases. The first scenario is the untrusted aggregation server case. Even if SMPC is used to protect the model updates, a malicious server may still be a problem in semi-honest models. In this case, TEE (like Intel SGX) can be used to provide protection for server-side operations. The second scenario is the existence of malicious client devices. Since client devices hold the data, they can see the model and can tamper with the protocol. Even if the devices behave benignly, external factors may create threats e.g., malicious mobile application can poison local training data or tamper updates. In these cases, TEEs (like ARM Trustzone) on the client-side can be used. In the worst-case scenario, when both server-side and client-side can be malicious, TEE can be employed on both sides [146]. Although a lot of work has been done in ML using TEEs, especially for cloud settings, very few works implement FL setting using TEE. Joint work with Intel and UPenn [147] used Intel SGX in FL setting for medical imaging. FL clients (medical institutions) locally train their data and send updated in encrypted form. The model is aggregated in SGX, and then it is sent to the clients in encrypted form. Both model data and data updates are protected in the scenario. In [146], Chen *et al.* used TEE for both client and server-side operations. On the other hand, they didn't provide a protection mechanism for the model updates, and they stated that updates are sent over the secure channel. In [148], only client-side operations are protected by TEE (Arm Trustzone), by partitioning the DNN model so that part of the model can be executed in TEE. For side-channel attacks, authors used Ohrimenko's approach, and for update protection, DP is used. The work was extended in [149] and in [150]. The work in [150] is the first attempt integrating TEEs both client and server side in FL setting. Authors used Arm Trustzone for client side and Intel SGX for server side. They conclude that the overhead introduced to client side is acceptable for the privacy improvements. This works also differ from [151] and [149] in term of DNN layer protection. The work aims to protect all DNN layers while previous works protect the most sensitive DNN layers leaving others unprotected. Therefore, it defends the model from model inversion, property and model inference attacks.

E. COMPARISON OF PRIVACY ENHANCING TECHNOLOGIES

We summarize the pros and cons of these privacy enhancing technologies and compare them. We considers the computation complexity, amount of data that needs to be transferred, communication round, accuracy level and the need for a special hardware as the performance metrics while comparing these PETs. Before starting the comparison of these technologies, we recall their functionalities briefly. HE allows execution of computation on encrypted data; SMPC enables parties to compute a joint function on their private data without revealing any additional information except the information learned from the output; DP introduces noise to the data for privacy but also allows computation on the data; TEE ensures the execution of computation in a secure and verifiable hardware environment. In terms of computation complexity, it is well known that the computation cost in HE is heavier than the others. The second costly solution is the SMPC where there may need some public key operations. The computation cost of usage of TEE is the least one because of not needing additional operations. In terms of communication data amount, HE and SMPC are not good candidates because of having long ciphertexts and additional data transfer needs, respectively. We don't see such costs in DP and TEE. Regarding the communication round overhead, some SMPC solutions may be considered as heavy because of needing more than two communication rounds. From the accuracy perspective, DP is the worst candidate because there is noise in the data in this type of solution while SMPC, HE and TEE outputs the same result when compared to the plain computation. Also, it should also be considered while comparing these solutions, TEE needs special hardware while others don't. This high-level comparison of the focused PETs and their functionalities are presented in Table 3.

In the context of privacy-preserving federated learning (PPFL), choosing the right privacy enhancing method is not straightforward because these methods are different in terms of effectiveness and computation cost. Using different metrics for evaluating data utility and data privacy can be a way to optimize the deployment of defense mechanisms [14]. In addition, since each privacy enhancing techniques have dominant advantages, combining different techniques may be useful to develop effective PPFL frameworks.

VII. GUIDELINES ON CHOOSING PETS

This section proposes a methodology for practitioners to select the right privacy approach for collaborative ML models. We first introduce the collaborative ML considerations and PETs constraints for different metrics. We start with discussing the collaborative ML model characteristics investigating the different angles. Then, we provide an analysis on the PETs constraints, mapping the appropriate PETs to the identified constraints. For each metric, we provide questions to be evaluated by the practitioners. Then we introduce a selection machinery based on evaluation of these questions as shown given in Figure 10.

A. COLLABORATIVE ML CONSIDERATIONS

1) COLLABORATION MODEL

What type of collaboration model and communication topology is employed by the parties?

Depending on the relations and interactions between the devices/parties, the collaboration model may change. Communication architecture is also tied to device interactions and affects the selection of collaborative ML model, e.g., what kind of communication protocol will be used and any resource (energy, compute, storage) limitation in place. These differences affect the PET selection in terms of what is shared during the training, the ownership of the model (or inference model), and the threat model, which will be analyzed separately.

What is shared among the parties involved in the training process changes with the collaboration model and impacts the communication and computation load. In FL and split learning, each client only exchanges the parameters with the server, so there is no communication between clients. In decentralized learning, each client exchanges with the neighboring client synchronously or asynchronously

TABLE 3. Functionality and comparison of PETs in terms of Computation Overhead (COMP), Communication Overhead (COMM), Communication Round Overhead (ROUND), Loss of Accuracy (ACCURACY LOSS), and requirement of a special hardware (HW).

	COMP	COMM	ROUND	ACCURACY LOSS	HW	Functionality
HE	High	High	No	Low	No	Computation on encrypted data
SMPC	Medium	High	Yes	Low	No	Joint computation on private data privately
DP	Low	Low	No	High	No	Computation on noised data
TEE	Low	Low	No	Low	Yes	Computation in secured environment

bringing more communication need. Therefore, SMPC solutions may be cumbersome in decentralized cases.

From the ownership of the model perspective, in FL and split learning cases, a server can serve the model for inference to the 3rd party users based on the business model. Like in cross-device FL, resulting model may be dispatched to other devices. In these cases, the model itself should be protected from malicious 3rd parties. Additionally, compromised devices may have white-box access to the model parameters. Hence, TEE solutions would be a good alternative to protect the model parameters from malicious 3rd party users. Since decentralized learning use cases mostly focus on local inference performed by participants themselves, malicious 3rd parties might not be an important concern.

2) ML ALGORITHM

How do the selected ML algorithm setting and operations impact the intended PET method?

In collaborative ML context, the distribution of data and collaboration models can affect the selection of ML algorithms. For horizontal and vertical settings, different algorithms may be chosen, and their efficiency may differ. A practitioner's choice of ML algorithm, collaboration model, and data distribution affect computation operations that need to be executed privately and so the selection of PET solutions. Algorithms that require extensive non-linear operations, usage of SMPC and HE might be costly compared to the DP. Additionally, the construction of efficient SMPC protocols depends on the ML algorithm's operations, and it is hard to provide an efficient generic algorithm while DP solutions are agnostic.

3) DATA DISTRIBUTION

How does the data distribution among parties impact the intended PET method?

The clients may have different data samples for the same feature space (called horizontal distribution), different feature spaces for the same data samples (called vertical distribution) or may have different data samples for different feature spaces (transfer learning). When data is distributed horizontally among clients, the local models can be trained using clients' local data with the same model architecture. All local parameters of the clients can simply be averaged to update the global model. In the case of vertical learning, methods like entity alignment techniques [152] are applied to gather common samples among parties which are then used to train machine learning models. Thus, the communication cost in the vertical split setting is increased in comparison with horizontal split. Also, a more complex mechanism is required to decompose the loss function at each client. Hence, the usage of methods like HE and DP, which have lower communication costs, might be preferred for vertical distributions.

The distribution of data with collaboration and ML models affects the computation operations that need to be executed privately and the selection of PET solutions. In horizontal case, which requires simple operations such as averaging, usage of somewhat or partially homomorphic encryption schemes are more practical compared to the fully homomorphic encryption schemes.

4) THREAT MODEL

What is the adversary model and his/her capabilities? How the data ownership and trust boundaries are defined for the parties?

Based on the identified collaboration model, data ownership, trust model, and adversary capabilities, the threat model can be identified to decide appropriate privacy solution. The adversary model should be defined for both server and clients, which can be trusted, honest-but-curious, or malicious. Honest-but-curious clients access the updated model parameters received from the server. In addition, malicious clients can tamper with the training process in the rounds they participate. Similarly, an honest-but-curious server may infer all client updates during the training process. In addition to this, a malicious server can tamper with the training process. In the scenarios where the server can be trusted for the computation of the global model, not trusted for the privacy of the clients, and the clients may be regarded as trusted entities, a PET solution such as secure aggregation may be enough. However, when the server has the potential to manipulate the global model in the direction of its intent, then more advanced solutions that make the clients ensure the correctness of the model need to be constructed, which will increase the overhead. Considering these types of threat models, possible PET solutions should be evaluated for whether they meet this requirement while keeping the computation and communication overhead as low as possible. Similar considerations are also needed for other threats, such as preventing the clients to provide non-legitimate updates for the model construction. The construction of privacy solutions with low overhead for honest-but-curious adversaries is more feasible than the solutions against malicious adversaries. As a result, the decision of the appropriate PET solution is highly dependent on the threat model.

5) DOMAIN SPECIFIC NEEDS

What are the domain specific needs, and how much does the intended PET method meet them?

As with the above-mentioned considerations, the domain characteristics of the data are important in applying AI/ML. Different use cases in vertical domains have different key performance indicators. For example, in mobile telecommunication domain, latency, energy efficiency, and secure handling and management of distributed data processing are important targets to achieve dependable, sustainable, and trustworthy networks. Hexa-X, which is a 6G flagship project, studies the pervasive use of AI/ML in future networks [153] and its privacy concerns. In healthcare and medical domain, accuracy, and processing of sensitive data in accordance with the regulations are critic aspects to consider, e.g., the accuracy concern prevents to integrate DP in medical data use cases, SMPC is preferred instead [154]. In the smart grid domain, consumption data is aggregated at different nodes, hence secure aggregation methods can be envisioned to increase efficiency.

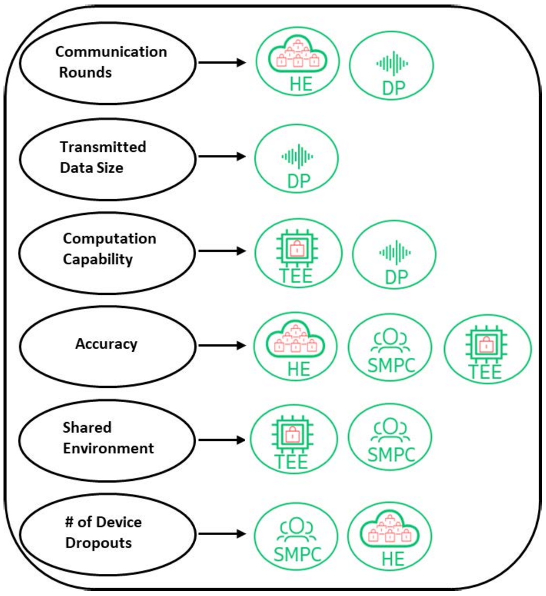

FIGURE 9. Constraints vs PETs mapping.

B. CONSTRAINTS

Selecting appropriate PETs also depends on constraints like computation capability and accuracy. For example, if there are constraints like accuracy or communication overhead, differential privacy or SMPC may not meet the concerns. If there is a need for distributed computation to improve the performance, this also may change the data ownership and trust boundaries since data or model will be distributed to compute nodes. In the following, we explain each constraint, as depicted in Figure 9, in detail. In addition to explaining the constraints, we provide recommendations about the selection of PET solutions with some selection motivations considering characteristics of PET solutions. We also depict this recommendation in Figure 9.

1) COMMUNICATION ROUNDS

How many extra data exchange rounds introduced by the intended PETs is tolerable for the execution of collaborative ML?

Communication rounds refer to the number of times data communicates between participants during a protocol or learning process. The number of communication rounds between parties is the main issue for collaborative learning tasks affecting the total communication overhead. Increasing communication efficiency is a challenging task and may cause redesigning the algorithms e.g., to reduce the communication cost of sending big weight matrices. If the bandwidth is limited and the concern is to have less communication rounds, then a privacy enhancing techniques that brings no overhead on the communication rounds, such as DP and HE, must be considered, where SMPC which may require to perform additional rounds might not be feasible. Since TEE solutions are mostly used to protect operations performed on the devices or attestation purposes, no additional communication cost is introduced. However, if attestation is used in TEE solution, additional communication rounds are needed.

2) TRANSMITTED DATA SIZE

How much data is allowed to be transferred between parties?

The amount of data transmitted among the parties during the protocol is an important efficiency parameter. In collaborative learning, the parameters might be updated and sent between parties several times. In SMPC solutions, there may be many communication rounds that increases the total amount of data needed to be transferred. In SMPC, the amount of data can also be very high because of the nature of the techniques. For example, in Yao's garbled circuits solution, which is a secure two-party computation protocol, to compute one-bit output of a function of two private bits of two parties, more than 768 bits must be transferred. When the transmitted data size is a constraint, DP can be preferred. TEEs do not put additional overhead in the network since their protection deals execution phase. If attestation is required in the system security design, additional steps are needed, hence latency concerns should be considered.

3) COMPUTATION CAPABILITY

How much computation resource is available to each party?

The need for computation demand increases as ML is adopted extensively a use case enabler. Bringing privacy also adds one more angle to this demand. Therefore, one needs to think about the available resources while investigating the appropriate privacy solution. When privacy preserving techniques are considered, their computation overhead should be considered as well. Considering the current privacy enhancing techniques, the computation

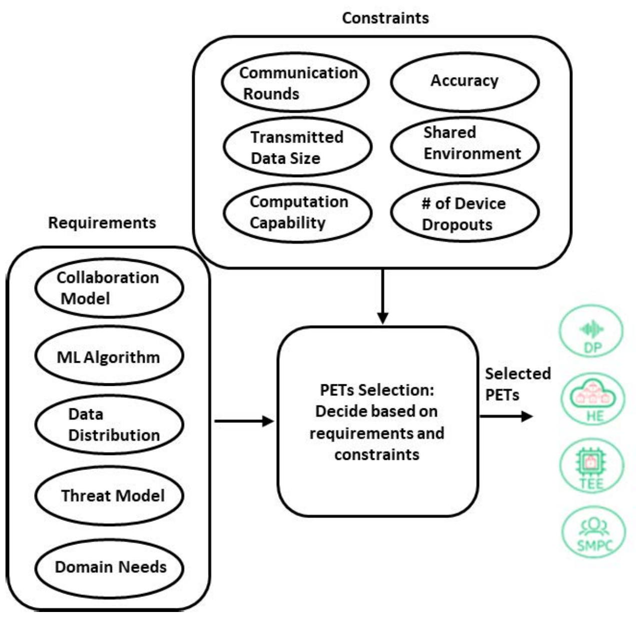

FIGURE 10. PET selection machinery.

overhead relationship, as discussed in Section VI.E, tends to order as such: HE > SMPC > DP. If the client devices have limited computation capability, the choice for privacy solution should be first DP then SMPC and HE, respectively. Bringing privacy using TEE depends on where TEE is available. If server-side protection e.g., protecting model privacy is the concern, then client-side limitations would not affect the choice. If client-side devices have the TEE capability, the natural choice to enable privacy would benefit from the feature. Depending on the client-side TEE capability, TEE can pave the way for protection against malicious client use cases.

4) ACCURACY

How much the accuracy loss could be allowed implementing the intended PET method.

In our context, accuracy can be defined as having minimum difference between the success probability of a model when there is a privacy solution is implemented or not. Since the differential privacy solutions add noise to the model parameters, the success probability of the model is decreased, which is called privacy versus utility trade-off. If accuracy is one of the critical parameters, then DP will not be the most appropriate privacy solution. Since homomorphic encryption allows computation on encryption of the actual data and not noise added data, the computation result will be the same as done in the computation on the plaintext data. Similar to homomorphic encryption, SMPC solutions allow the use of parties' private inputs to construct the model without revealing the private inputs. Thus, SMPC is another good candidate for collaborative ML when accuracy is very important. There may be some cases where to improve the performances by means of computation and communication, some tricks can be applied which may result in some information loss about data and decrease the accuracy. For example, truncation of weights in neural network can decrease the communication and computation cost but also decreases the accuracy. Using the TEE as part of the privacy solution does not affect the accuracy of the model.

5) SHARED ENVIRONMENT

Will the solution be deployed in a multi-tenant environment?

Concerns stemming from multi-tenant cloud environments also apply for ML use cases if a ML as a service (MaaS) cloud model is in place. For example, if FL server-side operations are performed on the cloud environment, then as in the other cloud-based services, memory isolation and shared environment access vulnerabilities will be important parameters. TEE is the only way forward to achieve protection for the data in memory. The most important aspect when using TEE is to partition the code into trusted and untrusted parts since they have memory and performance constraints [155]. The code should be partitioned in a secure way so that information disclosed outside of the enclave should not be used for adversarial purposes. Recent developments on library operating systems (lib-OS) pave the way for running applications without any decomposition effort. Although they are arguable for performance overhead and trusted computing base size, they are preferred for ease of development and comparable performance. Graphene SGX [156], Occlum [157], Fortanix [158], Anjuna [159], Scone [160] are some examples of lib-OSs that can be used to execute applications on Intel SGX. Another approach for the shared environment considerations could be dividing the data and operations between non-colluding servers, utilizing MPC techniques, into non-colluding servers running on different cloud environments.

6) THE NUMBER OF DEVICE DROPOUTS

Is there any possibility of devices getting dropout?

Especially in cross-device cases, some devices cannot participate in some iterations because of any intentional or unintentional reasons. To clarify the issue, the following example can be given for unintentional dropouts. To construct an ML model which is used for word prediction, many mobile phone users make contributions using users' wording behavior. In the model training phase, it is natural that some of the users may have some connection problems. For performance considerations, usage of small subsets of users can be considered, especially in federated learning. Usage of different subsets for each iteration can be given as an example for such an intentional dropout. Since secure multi-party computation requires no trusted party and enables execution of functions without revealing private inputs, this technique can overcome the dropout cases. Also, with the help of threshold cryptography it is not needed to have all the clients to join to the recovery of secrets, sensitive data, and decryption keys, which may help SMPC and HE solutions to handle the device dropouts during the execution of collaborative ML.

VIII. OPEN ISSUES AND FUTURE DIRECTIONS

In this section we listed some of the identified issues and research directions as follows.

- Collaborative ML and privacy enhancements have been heavily studied in the last decades, and new approaches have been proposed, like FL and SL. The main motivation of both approaches is enabling the generation of the global model from data without allowing the server to observe the data. The clients send some parameters computed from local data to the server to contribute to the construction of the global model. In FL, these parameters are the updates of the model, while in split learning, they are gradients of the cut layer. Although the clients don't send their data to the server, the parameters sent to the server may leak sensitive information about the data. To prevent this information leakage, researchers have proposed a considerable amount of work using privacy-enhancing technologies for FL in the literature. However, for the split learning case, privacy analysis and privacy enhancement studies are not enough to understand the remaining privacy issues in split learning yet.
- Also, considering the existing studies on privacy enhancements in collaborative ML, there is no significant effort on a systematic approach or a generic framework. Proposed solutions are mainly custom solutions that try to solve privacy issues from a specific point of view, not by a systematic approach. One possible future direction would be scrutinizing privacy issues on recent advances in collaborative ML and creating systematic approaches and frameworks both for analysis of privacy issues and solutions.
- Another possible study area would be the construction of hybrid solutions that use multiple privacy enhancing technologies instead of adopting only one of them. For example, using secure multi-party computation with confidential computing may be more feasible considering not only privacy but also performance point of view. Also, thinking of the trust relations, TEE may not solve all privacy considerations; complementing it with e.g., HE or SMPC may provide a stronger privacy guarantee.
- Verifiability of computations to provide proof points in collaborative ML/AI message flow is another important aspect for further studies. This can be considered from three angles in the collaborative scenarios: The first one is examining by the server-side to determine if the clients perform the intended computations in good faith. Second is examining by the client-side to determine if the server behaves in a good manner following the protocol steps. And the third is to prove that the environment on which server and client-side computations run has not been maliciously tampered. Different methodologies enable these kinds of proofs including formal cryptographic zero-knowledge proofs and attestation protocols based on TEEs.
- Last but not least, considering the different use cases and their constraints or requirements, a use-case specific framework can be provided to guide the privacy and industry practitioners. For example, the requirement for the IoT cases where resource constrained IoT devices are used would be different from the cross-silo use cases where different enterprises participate.

IX. CONCLUSION

As more data is generated and circulates in distributed devices, training the data in a centralized fashion is not feasible due to communication overhead and privacy implications. Therefore, ML methodologies with better efficiency and greater privacy considerations are needed where different actors and data owners involved in the training process. Collaborative ML approaches like federated learning pave the way for more privacy for the end users as they do not need to send raw data to any central server but rather contribute to the global model by local training. On the other hand, these approaches do not solve all privacy concerns, as studies show that model updates can leak private information. Fortunately, privacy enhancing technologies serve as a set of building blocks to remedy privacy issues.

In this study, different collaborative ML techniques and their challenges are explained. A detailed threat model, possible security and privacy attacks on these techniques are covered. Then, a comprehensive literature review of privacy preserving technologies applicable for collaborative ML is provided.

To the best of our knowledge, while there are taxonomies on collaborative ML techniques and attacks in literature, there is no guideline for choosing appropriate privacy enhancing techniques. To fill this gap, we introduce a guideline on how to approach PETs when designing a privacy-enhanced collaborative ML method taking the collaborative ML requirements and PETs' constraints into account. Additionally, possible research areas are discussed to shed light on further studies.

REFERENCES

[1] V. Berggren, R. Inam, L. Mokrushin, A. Hata, J. Jeong, S. Mohalik, J. Forgeat, and S. Sorrentino, "Artificial intelligence in next-generation connected systems," Ericsson Res., Ericsson White Paper GFTL-21:001105 Uen, Sep. 2021. [Online]. Available: https://www.ericsson.com/4a5720/assets/local/reports-papers/white-papers/09092021-artificial-intelligence-in-next-generation-connected-systems-whitepaper-v4.pdf

[2] H. Li, K. Li, J. An, and K. Li, "MSGD: A novel matrix factorization approach for large-scale collaborative filtering recommender systems on GPUs," *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 7, pp. 1530–1544, Jul. 2018.

[3] R. Guo, F. Zhang, L. Wang, W. Zhang, X. Lei, R. Ranjan, and A. Y. Zomaya, "BaPa: A novel approach of improving load balance in parallel matrix factorization for recommender systems," *IEEE Trans. Comput.*, vol. 70, no. 5, pp. 789–802, May 2021.

[4] F. Zhang, E. Xue, R. Guo, G. Qu, G. Zhao, and A. Y. Zomaya, "DS-ADMM++: A novel distributed quantized ADMM to speed up differentially private matrix factorization," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 6, pp. 1289–1302, Jun. 2022.

[5] *The Privacy Blog: Part 3, Privacy-Preserving Technologies*. Accessed: Aug. 1, 2022. [Online]. Available: https://the-privacy-blog.eu/2022/06/03/part-3-privacy-preserving-technologies/

[6] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction APIs," in *Proc. 25th USENIX Secur. Symp.*, 2016, pp. 601–618.

[7] N. Papernot, P. McDaniel, A. Sinha, and M. P. Wellman, "SoK: Security and privacy in machine learning," in *Proc. IEEE Eur. Symp. Secur. Privacy*, Apr. 2018, pp. 399–414.

[8] A. Chakraborty, M. Alam, V. Dey, A. Chattopadhyay, and D. Mukhopadhyay, "Adversarial attacks and defences: A survey," 2018, *arXiv:1810.00069*.

[9] B. Wang and N. Z. Gong, "Stealing hyperparameters in machine learning," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2018, pp. 36–52.

[10] V. Mothukuri, R. M. Parizi, S. Pouriyeh, Y. Huang, A. Dehghantanha, and G. Srivastava, "A survey on security and privacy of federated learning," *Future Gener. Comput. Syst.*, vol. 115, pp. 619–640, Feb. 2021.

[11] A. Blanco-Justicia, J. Domingo-Ferrer, S. Martínez, D. Sánchez, A. Flanagan, and K. E. Tan, "Achieving security and privacy in federated learning systems: Survey, research challenges and future directions," *Eng. Appl. Artif. Intell.*, vol. 106, Nov. 2021, Art. no. 104468.

[12] A. Boulemtafes, A. Derhab, and Y. Challal, "A review of privacy-preserving techniques for deep learning," *Neurocomputing*, vol. 384, pp. 21–45, Apr. 2020.

[13] Q. Li, Z. Wen, Z. Wu, S. Hu, N. Wang, Y. Li, X. Liu, and B. He, "A survey on federated learning systems: Vision, hype and reality for data privacy and protection," *IEEE Trans. Knowl. Data Eng.*, early access, Nov. 2, 2021, doi: 10.1109/TKDE.2021.3124599.

[14] X. Yin, Y. Zhu, and J. Hu, "A comprehensive survey of privacy-preserving federated learning: A taxonomy, review, and future directions," *ACM Comput. Surveys*, vol. 54, no. 6, pp. 1–36, Jul. 2021.

[15] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, and R. G. D'Oliveira, "Advances and open problems in federated learning," *Found. Trends Mach. Learn.*, vol. 14, nos. 1–2, pp. 1–210, Jun. 2021.

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

[17] C. Ma, J. Li, M. Ding, H. Yang, F. Shu, T. Suek, and H. V. Poor, "On safeguarding privacy and security in the framework of federated learning," *IEEE Netw.*, vol. 34, no. 4, pp. 242–248, Jul./Aug. 2020.

[18] P. Vepakomma, T. Swedish, R. Raskar, O. Gupta, and A. Dubey, "No peek: A survey of private distributed deep learning," 2018, *arXiv:1812.03288*.

[19] M. Rigaki and S. Garcia, "A survey of privacy attacks in machine learning," 2020, *arXiv:2007.07646*.

[20] L. Lyu, H. Yu, and Q. Yang, "Threats to federated learning: A survey," 2020, *arXiv:2003.02133*.

[21] D. Enthoven and Z. Al-Ars, "An overview of federated deep learning privacy attacks and defensive strategies," 2020, *arXiv:2004.04676*.

[22] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. 20th Int. Conf. Artif. Intell. Statist.*, 2017, pp. 1273–1282.

[23] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, and S. Ghemawat, "TensorFlow: Large-scale machine learning on heterogeneous distributed systems," 2016, *arXiv:1603.04467*.

[24] G. Xu, H. Li, S. Liu, K. Yang, and X. Lin, "VerifyNet: Secure and verifiable federated learning," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 911–926, 2020.

[25] T. Yang, G. Andrew, H. Eichner, H. Sun, W. Li, N. Kong, D. Ramage, and F. Beaufays, "Applied federated learning: Improving Google keyboard query suggestions," 2018, *arXiv:1812.02903*.

[26] Q. Yang, Y. Liu, Y. Cheng, Y. Kang, T. Chen, and H. Yu, "Federated learning," *Synth. Lectures Artif. Intell. Mach. Learn.*, vol. 13, no. 3, pp. 1–207, 2019.

[27] A. F. Karr, X. Lin, A. P. Sanil, and J. P. Reiter, "Privacy-preserving analysis of vertically partitioned data using secure matrix products," *J. Off. Statist.*, vol. 25, pp. 125–138, Jan. 2009.

[28] S. Hardy, W. Henecka, H. Ivey-Law, R. Nock, G. Patrini, G. Smith, and B. Thorne, "Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption," 2017, *arXiv:1711.10677*.

[29] Y. Liu, Y. Kang, X. Zhang, L. Li, Y. Cheng, T. Chen, M. Hong, and Q. Yang, "A communication efficient vertical federated learning framework," 2019, *arXiv:1912.11187*.

[30] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, Jan. 2021.

[31] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. Mcmahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for privacy-preserving machine learning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2017, pp. 1175–1191.

[32] Y. Liu, Y. Kang, C. Xing, T. Chen, and Q. Yang, "A secure federated transfer learning framework," *IEEE Intell. Syst.*, vol. 35, no. 4, pp. 70–82, Jul./Aug. 2020.

[33] S. Sharma, C. Xing, Y. Liu, and Y. Kang, "Secure and efficient federated transfer learning," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2019, pp. 2569–2576.

[34] P. Vepakomma, O. Gupta, T. Swedish, and R. Raskar, "Split learning for health: Distributed deep learning without sharing raw patient data," 2018, *arXiv:1812.00564*.

[35] C. Thapa, M. A. P. Chamikara, S. Camtepe, and L. Sun, "SplitFed: When federated learning meets split learning," 2020, *arXiv:2004.12088*.

[36] S. Abuadbba, K. Kim, M. Kim, C. Thapa, S. A. Camtepe, Y. Gao, H. Kim, and S. Nepal, "Can we use split learning on 1D CNN models for privacy preserving training?" in *Proc. 15th ACM Asia Conf. Comput. Commun. Secur.*, Oct. 2020, pp. 305–318.

[37] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-IID data," 2018, *arXiv:1806.00582*.

[38] Y. Lin, S. Han, H. Mao, Y. Wang, and W. J. Dally, "Deep gradient compression: Reducing the communication bandwidth for distributed training," 2017, *arXiv:1712.01887*.

[39] A. Singh, P. Vepakomma, O. Gupta, and R. Raskar, "Detailed comparison of communication efficiency of split learning and federated learning," 2019, *arXiv:1909.09145*.

[40] J. Vikström, "Comparing decentralized learning to federated learning when training deep neural networks under churn," M.S. thesis, School Elect. Eng. Comput. Sci. (EECS), KTH, Sweden, 2021.

[41] E. Tabassi, K. J. Burns, M. Hadjimichael, A. D. Molina-Markham, and J. T. Sexton, "A taxonomy and terminology of adversarial machine learning," NIST IR, Gaithersburg, MD, USA, Tech. Rep. NISTIR 8269, 2019, pp. 1–29.

[42] N. Akhtar and A. Mian, "Threat of adversarial attacks on deep learning in computer vision: A survey," *IEEE Access*, vol. 6, pp. 14410–14430, 2018.

[43] S. Huang, N. Papernot, I. Goodfellow, Y. Duan, and P. Abbeel, "Adversarial attacks on neural network policies," 2017, *arXiv:1702.02284*.

[44] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *Proc. ACM Asia Conf. Comput. Commun. Secur.*, Apr. 2017, pp. 506–519.

[45] C. Guo, J. Gardner, Y. You, A. G. Wilson, and K. Weinberger, "Simple black-box adversarial attacks," in *Proc. 36th Int. Conf. Mach. Learn.* (Proceedings of Machine Learning Research), vol. 97, K. Chaudhuri and R. Salakhutdinov, Eds. Jun. 2019, pp. 2484–2493. [Online]. Available: https://proceedings.mlr.press/v97/guo19a.html

[46] C. Hazay and Y. Lindell, "A note on the relation between the definitions of security for semi-honest and malicious adversaries," IACR Cryptol. ePrint Arch., vol. 2010, 2010, p. 551.

[47] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Federated learning of deep networks using model averaging," 2016, *arXiv:1602.05629*.

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

[49] B. Wang, Y. Yao, S. Shan, H. Li, B. Viswanath, H. Zheng, and B. Y. Zhao, "Neural cleanse: Identifying and mitigating backdoor attacks in neural networks," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2019, pp. 707–723.

[50] B. Chen, W. Carvalho, N. Baracaldo, H. Ludwig, B. Edwards, T. Lee, I. Molloy, and B. Srivastava, "Detecting backdoor attacks on deep neural networks by activation clustering," 2018, *arXiv:1811.03728*.

[51] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2020, pp. 2938–2948.

[52] Y. Jiang, Y. Li, Y. Zhou, and X. Zheng, "Mitigating sybil attacks on differential privacy based federated learning," 2020, *arXiv:2010.10572*.

[53] A. N. Bhagoji, D. Cullina, C. Sitawarin, and P. Mittal, "Enhancing robustness of machine learning systems via data transformations," in *Proc. 52nd Annu. Conf. Inf. Sci. Syst. (CISS)*, Mar. 2018, pp. 1–5.

[54] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Proc. 53rd Annu. Allerton Conf. Commun., Control, Comput. (Allerton)*, Sep. 2015, pp. 1310–1321.

[55] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2017, pp. 3–18.

[56] S. Truex, L. Liu, M. E. Gursoy, L. Yu, and W. Wei, "Demystifying membership inference attacks in machine learning as a service," *IEEE Trans. Services Comput.*, vol. 14, no. 6, pp. 2073–2089, Nov. 2021.

[57] S. Szyller, B. G. Atli, S. Marchal, and N. Asokan, "DAWN: Dynamic adversarial watermarking of neural networks," in *Proc. 29th ACM Int. Conf. Multimedia*, Oct. 2021, pp. 4417–4425.

[58] T. Orekondy, B. Schiele, and M. Fritz, "Prediction poisoning: Towards defenses against DNN model stealing attacks," 2019, *arXiv:1906.10908*.

[59] E. U. Soykan, Z. Bilgin, M. A. Ersoy, and E. Tomur, "Differentially private deep learning for load forecasting on smart grid," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, Dec. 2019, pp. 1–6.

[60] A. Wood, M. Altman, A. Bembenek, M. Bun, M. Gaboardi, J. Honaker, K. Nissim, D. O'Brien, T. Steinke, and S. Vadhan, "Differential privacy: A primer for a non-technical audience," *SSRN Electron. J.*, vol. 21, no. 1, pp. 209–275, 2018.

[61] C. Dwork and A. Roth, "The algorithmic foundations of differential privacy," *Found. Trends Theor. Comput. Sci.*, vol. 9, nos. 3–4, pp. 211–407, 2013.

[62] I. Mironov, "Rényi differential privacy," in *Proc. IEEE 30th Comput. Secur. Found. Symp. (CSF)*, Santa Barbara, CA, USA, Aug. 2017, pp. 263–275, doi: 10.1109/CSF.2017.11.

[63] H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang, "Learning differentially private recurrent language models," in *Proc. 6th Int. Conf. Learn. Represent.*, Vancouver, BC, Canada, 2018, pp. 1–14. [Online]. Available: https://openreview.net/forum?id=BJ0hF1Z0b

[64] R. Bassily, A. Smith, and A. Thakurta, "Private empirical risk minimization: Efficient algorithms and tight error bounds," in *Proc. IEEE 55th Annu. Symp. Found. Comput. Sci.*, Philadelphia, PA, USA, Oct. 2014, pp. 464–473, doi: 10.1109/FOCS.2014.56.

[65] N. Papernot, M. Abadi, U. Erlingsson, I. J. Goodfellow, and K. Talwar, "Semi-supervised knowledge transfer for deep learning from private training data," in *Proc. 5th Int. Conf. Learn. Represent.*, Toulon, France, 2017, pp. 1–16. [Online]. Available: https://openreview.net/forum?id=HkwoSDPgg

[66] C. Dwork, F. McSherry, K. Nissim, and A. D. Smith, "Calibrating noise to sensitivity in private data analysis," in *Proc. 3rd Theory Cryptograph. Conf. (TCC)* (Lecture Notes in Computer Science), vol. 3876. New York, NY, USA: Springer, Mar. 2006, pp. 265–284, doi: 10.1007/11681878_14.

[67] N. Agarwal, A. T. Suresh, F. X. Yu, S. Kumar, and B. McMahan, "cpSGD: Communication-efficient and differentially-private distributed SGD," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds. Montréal, QC, Canada, 2018, pp. 7575–7586. [Online]. Available: https://proceedings.neurips.cc/paper/2018/hash/21ce689121e39821d07d04faab328370-Abstract.html

[68] A. Bhowmick, J. Duchi, J. Freudiger, G. Kapoor, and R. Rogers, "Protection against reconstruction and its applications in private federated learning," 2018, *arXiv:1812.00984*.

[69] R. C. Geyer, T. Klein, and M. Nabi, "Differentially private federated learning: A client level perspective," 2017, *arXiv:1712.07557*.

[70] H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang, "Learning differentially private recurrent language models," 2017, *arXiv:1710.06963*.

[71] J. Hsu, S. Khanna, and A. Roth, "Distributed private heavy hitters," in *Automata, Languages, and Programming* (Lecture Notes in Computer Science), vol. 7391, A. Czumaj, K. Mehlhorn, A. M. Pitts, and R. Wattenhofer, Eds. Berlin, Germany: Springer, 2012, pp. 461–472, doi: 10.1007/978-3-642-31594-7_39.

[72] J. C. Duchi, M. Jordan, and M. J. Wainwright, "Local privacy and statistical minimax rates," in *Proc. IEEE 54th Annu. Symp. Found. Comput. Sci. (FOCS)*, Berkeley, CA, USA, Oct. 2013, pp. 429–438, doi: 10.1109/FOCS.2013.53.

[73] Ú. Erlingsson, V. Pihur, and A. Korolova, "RAPPOR: Randomized aggregatable privacy-preserving ordinal response," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Scottsdale, AZ, USA, G. Ahn, M. Yung, and N. Li, Eds. Nov. 2014, pp. 1054–1067, doi: 10.1145/2660267.2660348.

[74] *Learning With Privacy AT Scale*. Accessed: May 27, 2022. [Online]. Available: https://machinelearning.apple.com/2017/12/06/learning-with-privacy-at-scale.html

[75] B. Ding, J. Kulkarni, and S. Yekhanin, "Collecting telemetry data privately," in *Proc. Adv. Neural Inf. Process. Syst.*, Long Beach, CA, USA, I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, Eds. 2017, pp. 3571–3580. [Online]. Available: https://proceedings.neurips.cc/paper/2017/hash/253614bbac999b38b5b60cae531c4969-Abstract.html

[76] H. Shin, S. Kim, J. Shin, and X. Xiao, "Privacy enhanced matrix factorization for recommendation with local differential privacy," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 9, pp. 1770–1782, Sep. 2018, doi: 10.1109/TKDE.2018.2805356.

[77] X. Gu, M. Li, Y. Cheng, L. Xiong, and Y. Cao, "PCKV: Locally differentially private correlated key-value data collection with optimized utility," in *Proc. 29th USENIX Secur. Symp.*, S. Capkun and F. Roesner, Eds. 2020, pp. 967–984. [Online]. Available: https://www.usenix.org/conference/usenixsecurity20/presentation/gu

[78] Q. Ye, H. Hu, X. Meng, and H. Zheng, "PrivKV: Key-value data collection with local differential privacy," in *Proc. IEEE Symp. Secur. Privacy (SP)*, San Francisco, CA, USA, May 2019, pp. 317–331, doi: 10.1109/SP.2019.00018.

[79] N. Wang, X. Xiao, Y. Yang, J. Zhao, S. C. Hui, H. Shin, J. Shin, and G. Yu, "Collecting and analyzing multidimensional data with local differential privacy," in *Proc. IEEE 35th Int. Conf. Data Eng. (ICDE)*, Macao, China, Apr. 2019, pp. 638–649, doi: 10.1109/ICDE.2019.00063.

[80] J. Duchi, M. Wainwright, and M. Jordan, "Minimax optimal procedures for locally private estimation," 2016, *arXiv:1604.02390*.

[81] T. T. Nguyễn, X. Xiao, Y. Yang, S. C. Hui, H. Shin, and J. Shin, "Collecting and analyzing data from smart device users with local differential privacy," 2016, *arXiv:1606.05053*.

[82] X. Gu, M. Li, Y. Cao, and L. Xiong, "Supporting both range queries and frequency estimation with local differential privacy," in *Proc. IEEE Conf. Commun. Netw. Secur. (CNS)*, Washington, DC, USA, Jun. 2019, pp. 124–132, doi: 10.1109/CNS.2019.8802778.

[83] X. Gu, M. Li, L. Xiong, and Y. Cao, "Providing input-discriminative protection for local differential privacy," in *Proc. IEEE 36th Int. Conf. Data Eng. (ICDE)*, Dallas, TX, USA, Apr. 2020, pp. 505–516, doi: 10.1109/ICDE48307.2020.00050.

[84] Y. Zhao, J. Zhao, M. Yang, T. Wang, N. Wang, L. Lyu, D. Niyato, and K.-Y. Lam, "Local differential privacy-based federated learning for Internet of Things," *IEEE Internet Things J.*, vol. 8, no. 11, pp. 8836–8853, Jun. 2021, doi: 10.1109/JIOT.2020.3037194.

[85] S. Truex, L. Liu, K.-H. Chow, M. E. Gursoy, and W. Wei, "LDP-Fed: Federated learning with local differential privacy," in *Proc. 3rd ACM Int. Workshop Edge Syst., Anal. Netw.*, Heraklion, Greece, A. Y. Ding and R. Mortier, Eds. Apr. 2020, pp. 61–66, doi: 10.1145/3378679.3394533.

[86] L. Sun, J. Qian, and X. Chen, "LDP-FL: Practical private aggregation in federated learning with local differential privacy," in *Proc. 13th Int. Joint Conf. Artif. Intell.*, Montreal, QC, Canada, Z. Zhou, Ed. Aug. 2021, pp. 1571–1578, doi: 10.24963/ijcai.2021/217.

[87] M. Naseri, J. Hayes, and E. De Cristofaro, "Local and central differential privacy for robustness and privacy in federated learning," 2020, *arXiv:2009.03561*.

[88] Z. Sun, P. Kairouz, A. T. Suresh, and H. B. McMahan, "Can you really backdoor federated learning?" 2019, *arXiv:1911.07963*.

[89] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. Mcmahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for privacy-preserving machine learning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Dallas, TX, USA, B. M. Thuraisingham, D. Evans, T. Malkin, and D. Xu, Eds. Oct. 2017, pp. 1175–1191, doi: 10.1145/3133956.3133982.

[90] G. Ács and C. Castelluccia, "I have a DREAM (differentially private smart metering)," in *Information Hiding* (Lecture Notes in Computer Science), vol. 6958, T. Filler, T. Pevný, S. Craver, and A. D. Ker, Eds. Berlin, Germany: Springer, 2011, pp. 118–132 doi: 10.1007/978-3-642-24178-9_9.

[91] S. Goryczka and L. Xiong, "A comprehensive comparison of multiparty secure additions with differential privacy," *IEEE Trans. Dependable Sec. Comput.*, vol. 14, no. 5, pp. 463–477, Oct. 2017, doi: 10.1109/TDSC.2015.2484326.

[92] A. Cheu, A. D. Smith, J. R. Ullman, D. Zeber, and M. Zhilyaev, "Distributed differential privacy via shuffling," in *Advances in Cryptology—-EUROCRYPT* (Lecture Notes in Computer Science), vol. 11476, Y. Ishai and V. Rijmen, Eds. Darmstadt, Germany: Springer, 2019, pp. 375–403, doi: 10.1007/978-3-030-17653-2_13.

[93] U. Erlingsson, V. Feldman, I. Mironov, A. Raghunathan, K. Talwar, and A. Thakurta, "Amplification by shuffling: From local to central differential privacy via anonymity," in *Proc. 13th Annu. ACM-SIAM Symp. Discrete Algorithms*, T. M. Chan, Ed. San Diego, CA, USA, 2019, pp. 2468–2479, doi: 10.1137/1.9781611975482.151.

[94] A. Bittau, Ú. Erlingsson, P. Maniatis, I. Mironov, A. Raghunathan, D. Lie, M. Rudominer, U. Kode, J. Tinnes, and B. Seefeld, "Prochlo: Strong privacy for analytics in the crowd," in *Proc. 26th Symp. Operating Syst. Princ.*, Shanghai, China, Oct. 2017, pp. 441–459, doi: 10.1145/3132747.3132769.

[95] A. Cheu, A. Smith, J. Ullman, D. Zeber, and M. Zhilyaev, "Distributed differential privacy via shuffling," 2018, *arXiv:1808.01394*.

[96] M. Abadi, A. Chu, I. Goodfellow, H. B. Mcmahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Vienna, Austria, E. R. Weippl, S. Katzenbeisser, C. Kruegel, A. C. Myers, and S. Halevi, Eds. Oct. 2016, pp. 308–318, doi: 10.1145/2976749.2978318.

[97] C. A. Melchor, G. Castagnos, and P. Gaborit, "Lattice-based homomorphic encryption of vector spaces," in *Proc. IEEE Int. Symp. Inf. Theory*, Toronto, ON, Canada, F. R. Kschischang and E. Yang, Eds. Jul. 2008, pp. 1858–1862, doi: 10.1109/ISIT.2008.4595310.

[98] C. Gentry, "A fully homomorphic encryption scheme," Ph.D. dissertation, Dept. Comput. Sci., Stanford Univ., Stanford, CA, USA, 2009.

[99] L. Morris, "Analysis of partially and fully homomorphic encryption," Dept. Comput. Sci., Rochester Inst. Technol., Rochester, NY, USA, Tech. Rep., 2013, pp. 1–5.

[100] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Advances in Cryptology—EUROCRYPT* (Lecture Notes in Computer Science), vol. 1592, J. Stern, Ed. Prague, Czech Republic: Springer, 1999, pp. 223–238, doi: 10.1007/3-540-48910-X_16.

[101] J. Fan and F. Vercauteren, "Somewhat practical fully homomorphic encryption," IACR Cryptol. ePrint Arch., 2012, p. 144. [Online]. Available: http://eprint.iacr.org/2012/144

[102] X. Yi, R. Paulet, and E. Bertino, *Fully Homomorphic Encryption*. Cham, Switzerland: Springer, 2014, pp. 47–66, doi: 10.1007/978-3-319-12229-8_3.

[103] Y. Yang, X. Huang, X. Liu, H. Cheng, J. Weng, X. Luo, and V. Chang, "A comprehensive survey on secure outsourced computation and its applications," *IEEE Access*, vol. 7, pp. 159426–159465, 2019, doi: 10.1109/ACCESS.2019.2949782.

[104] C. Zhang, S. Li, J. Xia, W. Wang, F. Yan, and Y. Liu, "BatchCrypt: Efficient homomorphic encryption for cross-silo federated learning," in *Proc. USENIX Annu. Tech. Conf.*, A. Gavrilovska and E. Zadok, Eds. 2020, pp. 493–506. [Online]. Available: https://www.usenix.org/conference/atc20/presentation/zhang-chengliang

[105] C. Mouchet, J. Troncoso-Pastoriza, J.-P. Bossuat, and J.-P. Hubaux, "Multiparty homomorphic encryption from ring-learning-with-errors," *Proc. Privacy Enhancing Technol.*, vol. 2021, no. 4, pp. 291–311, Oct. 2021, doi: 10.2478/popets-2021-0071.

[106] A. López-Alt, E. Tromer, and V. Vaikuntanathan, "On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption," in *Proc. 44th Symp. Theory Comput. Conf.*, H. J. Karloff and T. Pitassi, Eds. New York, NY, USA: Association for Computing Machinery, 2012, pp. 1219–1234, doi: 10.1145/2213977.2214086.

[107] D. Mittal, D. Kaur, and A. Aggarwal, "Secure data mining in cloud using homomorphic encryption," in *Proc. IEEE Int. Conf. Cloud Comput. Emerg. Markets (CCEM)*, Oct. 2014, pp. 1–7.

[108] Y. Aono, T. Hayashi, L. T. Phong, and L. Wang, "Privacy-preserving logistic regression with distributed data sources via homomorphic encryption," *IEICE Trans. Inf. Syst.*, vol. 99, no. 8, pp. 2079–2089, 2016, doi: 10.1587/transinf.2015INP0020.

[109] L. T. Phong, Y. Aono, T. Hayashi, L. Wang, and S. Moriai, "Privacy-preserving deep learning via additively homomorphic encryption," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 5, pp. 1333–1345, May 2018, doi: 10.1109/TIFS.2017.2787987.

[110] Y. Chen, X. Qin, J. Wang, C. Yu, and W. Gao, "FedHealth: A federated transfer learning framework for wearable healthcare," *IEEE Intell. Syst.*, vol. 35, no. 4, pp. 83–93, Jul. 2020.

[111] H. Fang and Q. Qian, "Privacy preserving machine learning with homomorphic encryption and federated learning," *Future Internet*, vol. 13, no. 4, p. 94, Apr. 2021.

[112] J. Park and H. Lim, "Privacy-preserving federated learning using homomorphic encryption," *Appl. Sci.*, vol. 12, no. 2, p. 734, Jan. 2022.

[113] L. Zhang, Z. Zhang, and C. Guan, "Accelerating privacy-preserving momentum federated learning for industrial cyber-physical systems," *Complex Intell. Syst.*, vol. 7, no. 6, pp. 3289–3301, Dec. 2021.

[114] A. C.-C. Yao, "How to generate and exchange secrets," in *Proc. 27th Annu. Symp. Found. Comput. Sci. (SFCS)*, Oct. 1986, pp. 162–167.

[115] O. Goldreich, S. Micali, and A. Wigderson, "How to play any mental game, or a completeness theorem for protocols with honest majority," in *Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali*. New York, NY, USA: Association for Computing Machinery, 2019, pp. 307–328.

[116] M. O. Rabin, "How to exchange secrets with oblivious transfer," *IACR Cryptol. ePrint Arch.*, vol. 2005, p. 187, Jan. 2005.

[117] Y. Ishai, J. Kilian, K. Nissim, and E. Petrank, "Extending oblivious transfers efficiently," in *Advances in Cryptology—CRYPTO* (Lecture Notes in Computer Science). Berlin, Germany: Springer, 2003.

[118] E. Boyle, G. Couteau, N. Gilboa, Y. Ishai, L. Kohl, P. Rindal, and P. Scholl, "Efficient two-round OT extension and silent non-interactive secure computation," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2019, pp. 291–308.

[119] E. Shi, T.-H. H. Chan, E. G. Rieffel, R. Chow, and D. X. Song, "Privacy-preserving aggregation of time-series data," in *Proc. NDSS*, 2011.

[120] Y. Khazbak, T. Tan, and G. Cao, "MLGuard: Mitigating poisoning attacks in privacy preserving distributed collaborative learning," in *Proc. 29th Int. Conf. Comput. Commun. Netw. (ICCCN)*, Aug. 2020, pp. 1–9.

[121] F. Karakoç, M. Önen, and Z. Bilgin, "Secure aggregation against malicious users," in *Proc. 26th ACM Symp. Access Control Models Technol.*, Jun. 2021, pp. 115–124.

[122] T. D. Nguyen, P. Rieger, H. Yalame, H. Möllering, H. Fereidooni, S. Marchal, M. Miettinen, A. Mirhoseini, A.-R. Sadeghi, T. Schneider, and S. Zeitouni, "FLGUARD: Secure and private federated learning," *IACR Cryptol. ePrint Arch.*, vol. 2021, p. 25, Jan. 2021.

[123] H. Fereidooni, S. Marchal, M. Miettinen, A. Mirhoseini, H. Mollering, T. D. Nguyen, P. Rieger, A.-R. Sadeghi, T. Schneider, H. Yalame, and S. Zeitouni, "SAFELearn: Secure aggregation for private federated learning," in *Proc. IEEE Secur. Privacy Workshops (SPW)*, May 2021, pp. 56–62.

[124] Q. Li and G. Cao, "Efficient and privacy-preserving data aggregation in mobile sensing," in *Proc. 20th IEEE Int. Conf. Netw. Protocols (ICNP)*, Oct. 2012, pp. 1–10.

[125] P. Mohassel and Y. Zhang, "SecureML: A system for scalable privacy-preserving machine learning," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2017, pp. 19–38.

[126] D. Bogdanov, S. Laur, and J. Willemson, "Sharemind: A framework for fast privacy-preserving computations," *IACR Cryptol. ePrint Arch.*, vol. 2008, p. 289, Oct. 2008.

[127] Y. Zhang, G. Bai, X. Li, C. Curtis, C. Chen, and R. K. L. Ko, "PrivColl: Practical privacy-preserving collaborative machine learning," 2020, *arXiv:2007.06953*.

[128] M. Barni, C. Orlandi, and A. Piva, "A privacy-preserving protocol for neural-network-based computation," in *Proc. 8th Workshop Multimedia Secur.*, 2006, pp. 146–151.

[129] B. D. Rouhani, M. S. Riazi, and F. Koushanfar, "DeepSecure: Scalable provably-secure deep learning," in *Proc. 55th ACM/ESDA/IEEE Design Autom. Conf. (DAC)*, Jun. 2018, pp. 1–6.

[130] J. Liu, M. Juuti, Y. Lu, and N. Asokan, "Oblivious neural network predictions via MiniONN transformations," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2017, pp. 619–631.

[131] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," 2019, *arXiv:1902.04885*.

[132] W. Du, Y. S. Han, and S. Chen, "Privacy-preserving multivariate statistical analysis: Linear regression and classification," in *Proc. Int. Conf. Data Mining (SDM)*, 2004, pp. 222–233.

[133] Y. Li, Y. Duan, and W. Xu, "PrivPy: Enabling scalable and general privacy-preserving computation," 2018, *arXiv:1801.10117*.

[134] P. Mohassel and P. Rindal, "ABY^3: A mixed protocol framework for machine learning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2018, pp. 35–52.

[135] A. Gascón, P. Schoppmann, B. Balle, M. Raykova, J. Doerner, S. Zahur, and D. Evans, "Privacy-preserving distributed linear regression on high-dimensional data," *Proc. Privacy Enhancing Technol.*, vol. 2017, no. 4, pp. 345–364, 2017.

[136] A. Nilsson, P. N. Bideh, and J. Brorsson, "A survey of published attacks on Intel SGX," 2020, *arXiv:2006.13598*.

[137] C. Fung, C. J. M. Yoon, and I. Beschastnikh, "Mitigating sybils in federated learning poisoning," 2018, *arXiv:1808.04866*.

[138] T. Lee, Z. Lin, S. Pushp, C. Li, Y. Liu, Y. Lee, F. Xu, C. Xu, L. Zhang, and J. Song, "Occlumency: Privacy-preserving remote deep-learning inference using SGX," in *Proc. 25th Annu. Int. Conf. Mobile Comput. Netw.*, Oct. 2019, pp. 1–17.

[139] T. Hunt, C. Song, R. Shokri, V. Shmatikov, and E. Witchel, "Chiron: Privacy-preserving machine learning as a service," 2018, *arXiv:1803.05961*.

[140] F. Tramèr and D. Boneh, "Slalom: Fast, verifiable and private execution of neural networks in trusted hardware," 2018, *arXiv:1806.03287*.

[141] Z. Gu, H. Jamjoom, D. Su, H. Huang, J. Zhang, T. Ma, D. Pendarakis, and I. Molloy, "Reaching data confidentiality and model accountability on the CalTrain," in *Proc. 49th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw. (DSN)*, Jun. 2019, pp. 336–348.

[142] N. Hynes, R. Cheng, and D. Song, "Efficient deep learning on multi-source private data," 2018, *arXiv:1807.06689*.

[143] K. G. Narra, Z. Lin, Y. Wang, K. Balasubramaniam, and M. Annavaram, "Privacy-preserving inference in machine learning services using trusted execution environments," 2019, *arXiv:1912.03485*.

[144] H. Hashemi, Y. Wang, and M. Annavaram, "DarKnight: A data privacy scheme for training and inference of deep neural networks," 2020, *arXiv:2006.01300*.

[145] L. Ng, S. S. M. Chow, A. P. Y. Woo, D. P. H. Wong, and Y. Zhao, "Goten: Gpu-outsourcing trusted execution of neural network training," in *Proc. AAAI Conf. Artif. Intell.*, 2021, pp. 14876–14883.

[146] Y. Chen, F. Luo, T. Li, T. Xiang, Z. Liu, and J. Li, "A training-integrity privacy-preserving federated learning scheme with trusted execution environment," *Inf. Sci.*, vol. 522, pp. 69–79, Jun. 2020.

[147] *Federated Learning for Medical Imaging*. Accessed: May 27, 2022. [Online]. Available: https://www.intel.com/content/www/us/en/artificial-intelligence/posts/federated-learning-for-medical-imaging.html#gs.z2vjqn

[148] F. Mo and H. Haddadi, "Efficient and private federated learning using tee," in *Proc. EuroSyst. Conf.*, 2019.

[149] F. Mo, A. S. Shamsabadi, K. Katevas, S. Demetriou, I. Leontiadis, A. Cavallaro, and H. Haddadi, "DarkneTZ: Towards model privacy at the edge using trusted execution environments," in *Proc. 18th Int. Conf. Mobile Syst., Appl., Services*, Jun. 2020, pp. 161–174.

[150] F. Mo, H. Haddadi, K. Katevas, E. Marin, D. Perino, and N. Kourtellis, "PPFL: Privacy-preserving federated learning with trusted execution environments," in *Proc. 19th Annu. Int. Conf. Mobile Syst., Appl., Services*, Jun. 2021, pp. 94–108.

[151] Z. Gu, H. Huang, J. Zhang, D. Su, H. Jamjoom, A. Lamba, D. E. Pendarakis, and I. Molloy, "Yerbabuena: Securing deep learning inference data via enclave-based ternary model partitioning," 2018, *arXiv:1807.00969*.

[152] Z. Yan, L. Guoliang, and F. Jianhua, "A survey on entity alignment of knowledge base," *J. Comput. Res. Develop.*, vol. 53, p. 165, Jan. 2016.

[153] F. Miltiadis, L. Vasiliki, M. Jafar, M. Mattia, E. U. Soykan, B. Tamas, R. Nandana, R. Nuwanthika, L. L. Magoarou, P. Pietro, and B. Andras, "Pervasive artificial intelligence in next generation wireless: The Hexa-X project perspective," in *Proc. 1st Int. Workshop Artif. Intell. Beyond 5G 6G Wireless Networks*, 2022.

[154] A. B. Ünal, N. Pfeifer, and M. Akgün, "CECILIA: Comprehensive secure machine learning framework," 2022, *arXiv:2202.03023*.

[155] T. D. Ngoc, B. Bui, S. Bitchebe, A. Tchana, V. Schiavoni, P. Felber, and D. Hagimont, "Everything you should know about Intel SGX performance on virtualized systems," in *Proc. ACM Meas. Anal. Comput. Syst.*, Jun. 2019, pp. 1–21.

[156] C. C. Tsai, D. E. Porter, and M. Vij, "Graphene-SGX: A practical library OS for unmodified applications on SGX," in *Proc. USENIX Annu. Tech. Conf.*, 2017, pp. 645–658.

[157] Y. Shen, H. Tian, Y. Chen, K. Chen, R. Wang, Y. Xu, Y. Xia, and S. Yan, "Occlum: Secure and efficient multitasking inside a single enclave of Intel SGX," in *Proc. 24th Int. Conf. Architectural Support Program. Lang. Operating Syst.*, Mar. 2020, pp. 955–970.

[158] J. G. Beekman and D. E. Porter, "Challenges for scaling applications across enclaves," in *Proc. 2nd Workshop Syst. Softw. Trusted Execution*, Oct. 2017, pp. 1–2.

[159] *Anjuna Confidential Cloud Software*. Accessed: May 27, 2022. [Online]. Available: https://www.anjuna.io/

[160] S. Arnautov, B. Trach, F. Gregor, T. Knauth, A. Martin, C. Priebe, J. Lind, D. Muthukumaran, D. O'Keeffe, M. Stillwell, D. Goltzsche, D. Eyers, R. Kapitza, P. R. Pietzuch, and C. Fetzer, "SCONE: Secure Linux containers with Intel SGX," in *Proc. OSDI*, 2016, pp. 689–703.

ELIF USTUNDAG SOYKAN received the M.S. and Ph.D. degrees in computational science and engineering from Istanbul Technical University. She had worked at The Scientific and Technological Research Council, National Cryptology Institute, for 13 years in security domain. She joined Ericsson Research, in 2018, where she is currently working as a Master Security Researcher. She has published several papers in international conferences, mostly on information security and privacy. Her research interests include ML/AI security, privacy enhancing technologies, and the IoT security.

LEYLI KARAÇAY received the M.Sc. and Ph.D. degrees in computer science and engineering from Sabanci University, Türkiye, in 2012 and 2020, respectively. She had worked as a Teaching Assistant at Sabanci University for seven years. She joined Ericsson Research, Türkiye, in October 2019. She currently works as an Experienced Security Researcher at Ericsson Research. She has some scientific research papers published in journals and conference proceeding in the area of security.

FERHAT KARAKOÇ received the B.Sc., M.Sc., and Ph.D. degrees in computer engineering from Istanbul Technical University. He has been working as a Security Researcher and 3GPP SA WG3 Delegate at Ericsson, since 2020. Before joining to Ericsson, he worked on information security and cryptography at private sector companies and public institutes. He also taught information security and cryptography related courses at universities. He has several scientific research papers on cryptography, published in journals and conferences.

EMRAH TOMUR received the B.S. and M.Sc. degrees in electronics engineering from Bilkent University, in 1999 and 2001, respectively, and the Ph.D. degree in information systems from Middle East Technical University, in 2008. He has been working as a Master Researcher at Ericsson, since January 2019, where he is heading the research teams in the area of security and network. Before joining Ericsson, he worked as an Research and Development Manager in private sector companies and a Technology Transfer Manager in universities, where he gave courses, and worked as a Graduate Thesis Advisor. He has several scientific research papers published in journals and conference proceedings in the area of security. He also worked in numerous various national and international research and development projects funded by EU or national agencies. His technical expertise is on security of the Internet of Things, M2M, and wireless sensor networks.