SoK: Secure Aggregation based on cryptographic schemes for Federated Learning

Mohamad Mansouri, Melek Önen, Wafa Ben Jaballah, Mauro Conti

SoK: Secure Aggregation based on cryptographic schemes for Federated Learning

Mansouri Mohamad
mohamad.mansouri@eurecom.com
Thales SIX GTS/ EURECOM
France

Melek Önen
melek.onen@eurecom.com
EURECOM
France

Wafa Ben Jaballah
wafa.benjaballah@thalesgroup.com
Thales SIX GTS
France

Mauro Conti
conti@math.unipd.it
University of Padua
Italy

ABSTRACT

Secure aggregation consists of computing the sum of data collected from multiple sources without disclosing these individual inputs. Secure aggregation has been found useful for various applications ranging from electronic voting to smart grid measurements. Recently, federated learning emerged as a new collaborative machine learning technology to train machine learning models. In this work, we study the suitability of secure aggregation based on cryptographic schemes to federated learning. We first provide a formal definition of the problem and suggest a systematic categorization of existing solutions. We further investigate the specific challenges raised by federated learning and analyze the recent dedicated secure aggregation solutions based on cryptographic schemes. We finally share some takeaway messages that would help a secure design of federated learning and identify open research directions in this topic. Based on the takeaway messages, we propose an improved definition of secure aggregation that better fits federated learning.

KEYWORDS

Secure Aggregation, Homomorphic Encryption, Multi-Party Computation, Federated Learning

1 INTRODUCTION

With the recent advances in information technology, the adoption of distributed systems become a major trend for various applications/systems such as electronic voting [KSRW04], smart grids [ADMC17], industry's actuators and sensors, swarms of drones etc. All these systems involve multiple geo-dispersed edge devices that usually collect, store and collaboratively aggregate data. Recently, federated learning emerged as a new collaborative machine learning technology to train machine learning models. This new technology consists of several federated clients holding private data and contributing to the training of a machine learning model in a collaborative manner: Each client first trains a local machine learning model using its own dataset and further shares training parameter results with a server. The server aggregates the training results and sends the aggregate back to the clients. The clients repeat the process so that they finally converge to one shared, global model. The global model's parameters consist of the average of each local model's parameters. Thanks to this new technology, clients can train more accurate machine learning models using larger and more diverse datasets (originating from other clients).

Consequently, aggregating data from multiple sources is never more useful as in these days (such as their use in federated learning). Nevertheless, such operations that usually involve sensitive data raise serious privacy concerns and call for suitable privacy enhancing technologies to protect the confidentiality of clients' data while still being able to perform the aggregation operation. During the past 20 years, a huge amount of research focused on designing secure aggregation solutions [ÖM07, LEM14a, BSK+19, BIK+17, DA16, CMT05] for various applications that enable the computation of the sum of several parties' inputs without leaking any information about each individual input except the aggregate (the sum). Secure aggregation can be achieved using four main privacy enhancing technologies: differential privacy, Trusted-Execution Environment (TEE), secure shuffling under anonymity assumptions, and cryptography.

In this paper, we propose to study secure aggregation solutions based on cryptographic schemes as these seem to be the most popular ones for federated learning (37 solutions). We believe that this is mainly due to their practical setup and to the fact they do not affect the accuracy of the results.We first identify and investigate the three phases of secure aggregation protocol: Setup, Protection, and Aggregation. We categorize existing solutions based on two cryptographic primitives: encryption and multi-party computation (MPC). We further investigate the suitability of these solutions to federated learning and identify additional security, privacy and performance challenges, accordingly. More specifically, we study whether and how encryption-based or MPC-based secure aggregation can cope with: client failures/dynamics, scalability in terms of clients' input size, scalability in terms of number of clients, the presence of malicious clients or malicious server, and statistical attacks that leak private information.

To summarize, our contributions are as follows:

- We propose a formal definition of secure aggregation based on cryptographic schemes, by identifying a set of features that help an effective comparison among existing solutions.
- We study the existing secure aggregation solutions based based on the underlying cryptographic schemes that are masking, additive homomorphic encryption, functional encryption, and multi-party computation. We analyze each

PoPETS, July 10–14, 2023, Lausanne, Switzerland
2023. https://doi.org/XXXXXXX.XXXXXXX

category with respect to the set of features we have identified.
- We further present a comprehensive study on the suitability of secure aggregation based on cryptographic schemes for federated learning, by identifying the main security, privacy, and performance challenges raised by this integration and their impact on each of the proposed categories.
- We review the 37 solutions that integrate secure aggregation for federated learning and we propose a categorization based on two axes: the challenges that each solution tackles and the type of secure aggregation used.
- We finish with some key takeaways that can help developers design, develop or improve secure aggregation schemes for federated learning.

The remaining of the paper is organized as follows: Section 2 presents the scope of our study, a historical background on SA based on cryptographic schemes and studies the threat model. Section 3 proposes a formal definition of SA and establishes a categorization of existing schemes. Section 4 focuses on the deployment of SA based on cryptographic schemes in federated learning. Furthermore, section 5 describes the major takeaways of this research. Finally, section 6 concludes this paper.

2 BACKGROUND AND MOTIVATION

This section briefly introduces federated learning and explains the need for secure aggregation for such applications. Then, we define the scope of our study by focusing our research on secure aggregation based on cryptographic schemes that are appropriate for federated learning. Third, we present a historical overview of secure aggregation. Last but not least, we provide the threat model and the security requirements.

2.1 Federated Learning

The term *Federated Learning* was initially introduced by McMahan et al. [MMR+17] and refers to a technology that enables training machine learning models on data from different sources without the need to store the data at a central location. Federated Learning (FL) is performed in several rounds with multiple clients and a server. A FL client is installed at each data source location. At the beginning, the FL server initiates the same model for all clients. For each FL round, the clients perform local training on their own data to improve the received machine learning model and send the updated model to the server. The latter aggregates the trained models received from clients by averaging them and then sends the outcome back to the clients. After the clients receive the aggregated model, a new FL round starts where the clients and the server repeat the steps. FL stops when the aggregated model converges.

The main goal of FL is to protect the privacy of the local data while still being able to use them for training public models. This technology provides a great advantage over other techniques that try to achieve the same goal (eg., training on encrypted data [VNP+20, HTGW18, WGC19, DGBL+16, WTB+20]). The latter adds a large computational overhead since it involves encryption of the inputs then performing complex computations on encrypted data. FL requires less computation as it only involves the averaging operation at the server.

While FL is proposed for privacy preserving purposes, it lacks formal guarantee of privacy. For example, adversaries who have access to the training results sent from each client to the server might be able to infer a training sample from a client's private dataset. Many types of inference attacks on FL are investigated and researched in [MSDCS19, ZLH19, LHCH20, NSH19]. To prevent such attacks, SA is used as a protection of the client's input while permitting the computation of the sum of all the updates.

2.2 The Scope of the Study

In this paper, aggregation refers to the process where data collected from multiple sources are summed up. Usually, data is provided in consecutive time periods and the sum (aggregate) is calculated for each time period. For many applications including federated learning, the data contain sensitive information.

Usually, SA involves two actors: *Users* ($\mathcal{U}$) and *Aggregators* ($\mathcal{A}$). *Users* are parties that provide the input data while *Aggregators* are the parties that perform the aggregation to obtain the sum. Additionally, a *Trusted Third Party* ($\mathcal{TP}$) may also be defined for setup purposes. There are mainly four types of secure aggregation: SA based on differential privacy, SA based on Trusted-Execution Environment (TEE), SA based on anonymity, and SA based on cryptography. In this paper, we specifically focus on SA based on cyrptographic schemes. In the following, we elaborate on each type of secure aggregation and we clarify the choice of studying SA based on cryptographic schemes.

SA based on differential privacy. Differential privacy (DP) is a technique that theoretically ensures that a change in a single sample in the training dataset of an algorithm, causes only statistically insignificant changes to the algorithm's output. To achieve secure aggregation using DP mechanisms, clients protect their trained models by adding statistical noise[MASN21, ASY+18, TF19, YWW21, CYD20, STL20, DHCP21]. The main concern with this approach is that the added noise is amplified during the aggregation thus, significantly affecting the accuracy of the aggregated model.

SA based on trusted-execution environment. Another method to perform secure aggregation is to use a TEE at the server [ZJF+21, NMZ+21, ZWC+21, MHK+21]. The TEE is a hardware-separated area of the main processor that guarantees the confidentiality and integrity of the processed data. SA can be achieved by computing the aggregation within the TEE. This method requires strong trust assumptions regarding the use of the TEE.

SA based on anonymity. A different method relies on anonymous communication assumption [IKOS06]. This method is also known as secure shuffling where users split their inputs into shares and send them anonymously to the aggregator [ZWC+21]. Thanks to the anonymous communication, the server cannot discover the origin of the received shares and thus is unable to reconstruct the user inputs. In the case of federated learning applications, it is often impossible to have completely anonymous communication channels between the clients and the server.

SA based on cryptography. Finally, cryptographic techniques are used to design secure aggregation protocols. Many researchers are particularly interested in this type of secure aggregation (even much

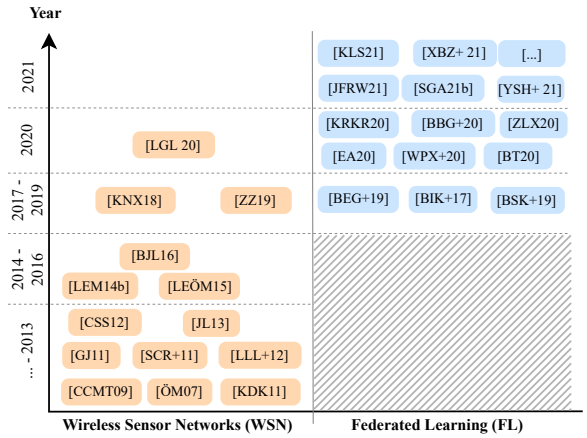

Figure 1: The publications that used terms "secure aggregation"and "privacy-preserving aggregation" in their title or abstract. The publications are grouped based on their application domain.

before federated learning existed) since it does not significantly affect the accuracy of the aggregation result. It also does not rely on impractical assumptions such as in the case of TEE and secure shuffling which makes this choice suitable for federated learning applications. Indeed, we count around around 37 solutions based on cryptographic schemes that are used for federated learning.

2.3 History of Secure Aggregation

SA was originally designed to aggregate private data generated by nodes in wireless sensor networks (WSN) (also in smart grids and smart meters applications). As far as we know, the "secure aggregation" term first appeared around 2003 [HE03]. With the increase in popularity of WSN and the Internet of Things (IoT), more solutions were proposed [ÖM07, CMT05]. It appeared also with alternative names that refer to the same concept. Namely, "privacy-preserving aggregation" [SCR+11], "privacy-friendly aggregation" [KDK11], and "private stream aggregation" [CSS12]. Later in 2017, after the emerge of the federated learning technology, Bonawitz et al. [BIK+17] identified the need for secure aggregation solutions for federated learning and proposed the first secure aggregation solution in this context. The scheme was an improvement and adaptation of an existing SA scheme [ÁC11]. After the first solution of SA for federated learning appeared in 2017 [BIK+17], more researchers further explored this area and propose more SA solutions for federated learning. Most of the proposed solutions are enlightened by existing solutions which were originally designed for WSN. A study on the appearance of the term "secure aggregation" and its alternatives is shown in Figure 1. The figure shows a clear research trend on secure aggregation in the context of federated learning.

2.4 Threat Model and Security Requirements

A common security model for secure aggregation schemes is the *honest-but-curious* model with *collusions*. An honest-but-curious party is a party that correctly follows the protocol steps but remains curious to discover any private information. Honest-but-curious users may collude with each other or/and with the honest-but-curious aggregator. In case of collusion, the colluding parties share all their private information. In this model, a SA solution is said to be secure if it achieves Aggregator Obliviousness.

Definition 1. ***Aggregator Obliviousness**: This security notion ensures that an honest-but-curious aggregator cannot learn more than what could be learned from the sum of the users inputs. If some users are corrupted (i.e., users sharing their private information with the aggregator), the notion only requires that the aggregator gets no extra information about the values of the honest users beyond that their aggregate value. This notion was initially proposed by Shi et al. [SCR+11] for SA schemes in the context of WSN.*

In addition to the *honest-but-curious* model, several research works considered a *malicious* model where the adversaries are more powerful. Malicious entities may alter their inputs to the protocol to either learn private information or to alter the result of the aggregation. There are two main malicious settings: The *malicious aggregator* model and the *malicious users* model. A SA solution is said to be secure if it achieves Aggregate Integrity and Aggregator Obliviousness.

Definition 2. ***Aggregate Integrity**: This security notion ensures that a malicious aggregator cannot forge a false aggregation result without being detected by the users. A forgery is defined as an aggregation outcome that is not equal to the sum of the users' inputs. Additionally, a set of malicious users cannot compromise the aggregation result as long as their is a sufficient number of honest users (more than a threshold). A compromise is defined as a significant drift in the value of the aggregation outcome from the sum of the honest users' inputs.*

In this paper, we first study SA solutions based on cryptographic schemes in the honest-but-curious model. Then, in section 4, we show how these schemes are extended to consider the malicious model adhering to federated learning use-cases.

3 SECURE AGGREGATION BASED ON CRYPTOGRAPHIC SCHEMES

In this section, we systematize crypto-based secure aggregation by first proposing a general definition that captures its different instantiations. We define it by describing three consecutive phases. Then, we regroup secure aggregation (SA) into two categories based on the underlying cryptographic tools used to build the SA protocol. Specifically, we distinguish encryption-based secure aggregation and multi-party-computation (MPC)-based secure aggregation. For each of the categories, we show how to build a secure aggregation protocol from the existing cryptographic schemes by defining the algorithms executed at each SA phase. We summarize the advantages and disadvantages of the different categories in Table 1.

3.1 Secure Aggregation Protocol Phases

A secure aggregation protocol consists of three consecutive phases: **SA.Setup**, **SA.Protect**, and **SA.Agg**. Each of these phases achieves a specific task described as follows:

- **SA.Setup**: In this phase, the n users and the aggregator get the public parameters and the key material. The public parameters and the keys are generated either using a trusted

	Schemes	$\mathcal{TP}$ required	No SC required	Dynamic users	Comp.	Comm.
Encryption	**Masking** (DC-net)	PKI	●	○	$O(n+m)$	$O(n+m)$
	AHE	KD	●	○	$O(m)$	$O(m)$
	FE	KD	●	●	$O(m)$	$O(m)$
MPC	**n-out-of-n SS** **t-out-of-n SS**	-	○	●	$O(nm)$	$O(nm)$

Table 1: Table comparing the different categories of secure aggregation. $\mathcal{TP}$ stands for trusted third party. SC stands for pre-established secure channels. Dynamic users property shows wether the aggregation can be performed with only a subset of the users. Comp. stands for computation cost on users. Comm. stands for communication cost between users and aggregators. n and m represents the number of users and the size of user's input respectively. ● means that the property is attained.

third party ($\mathcal{TP}$) or through a distributed mechanism. At the end of this phase, each user stores a single unique key k_i where $i \in [1, .., n]$ and the aggregator stores its aggregation key k_0.

- **SA.Protect**: Each user $\mathcal{U}_i$ locally executes a protection algorithm to protect its input $m_{i,\tau}$ of time period τ. The resulting protected input is sent to the aggregator(s).
- **SA.Agg**: After the aggregators collect all the protected inputs, they collaboratively execute an aggregation algorithm to retrieve the sum of user inputs for time period τ. In the case with a single aggregator, the aggregation algorithm is locally executed by the aggregator.

3.2 Encryption-based SA

Encryption-based SA protocols use encryption schemes to protect the inputs of the users. Encryption utilizes a secret key to ensure the confidentiality of the user input. However, to further achieve *Aggregator Obliviousness* (see Definition 1), users should not be allowed to encrypt their inputs with the same key. Moreover, the encryption scheme should allow the computation of the sum of the inputs over the ciphertexts without leaking the individual cleartext values. There are three types of encryption schemes that are used to build a secure aggregation protocol: (i) masking, (ii) additively homomorphic encryption (AHE), and (iii) functional encryption (FE). In general, encryption-based SAs rely on a single aggregator to perform the aggregation which minimizes the communication overheard of the protocol.

3.2.1 ***Secure Aggregation using Masking.*** Masking is a symmetric encryption technique based on one-time pad [Rub96]. It uses modular addition to mask the data owner inputs. Given a shared key k between two parties and an upper bound r of the message, masking is defined by two algorithms:

- $c \leftarrow Masking.Mask(k, m)$: Masks an input m with the masking key k ($c = m + k \mod r$).
- $m \leftarrow Masking.UnMask(k, c)$: Unmasks the ciphertext c using the masking key k ($c = c - k \mod r$).

It is one of the oldest techniques for designing a secure aggregation protocol. It was used first in tree structured networks. These schemes are called layered masking schemes [CMT05, ÖM07, CCMT09]. We describe an example of these schemes in Appendix B. More recently, a Dining Cryptographers network (DC-net) variant is proposed in [ÁC11, BIK+17, BBG+20, SGA21b].

In the **SA.Setup** phase, each pair of users $(\mathcal{U}_i, \mathcal{U}_j)$ agrees on a random key $k_{(i,j),\tau}$ using a *Key Agreement* protocol (ex., Diffie Hellman (DH) [DH06] using the aggregator as a proxy to forward the public keys). Also, each user $\mathcal{U}_i$ agrees on a random key $k_{(i,0),\tau}$ with the aggregator. As a result, each user $\mathcal{U}_i$ and the aggregator $\mathcal{A}$ computes their own unique key as follows:

$$k_{i,\tau} \leftarrow \sum_{j=1}^{i-1} k_{(i,j),\tau} - \sum_{j=i+1}^{n} k_{(i,j),\tau} - k_{(i,0),\tau} \quad \text{s.t.} \quad k_{(i,j),\tau} = k_{(j,i),\tau} \text{ and } k_{(i,i),\tau} = 0 \quad \forall\, i \in [0, .., n] \tag{1}$$

In the **SA.Protect** phase, each user masks its own input $m_{i,\tau}$ with the key $k_{i,\tau}$:

$$c_{i,\tau} \leftarrow Masking.Mask(k_{i,\tau}, m_{i,\tau})$$

In the **SA.Agg** phase, the aggregator adds the masked inputs from all users. Then, it removes the mask using its key $k_{0,\tau}$ (all the operation are mod $R = nR_u$ where $[0, R_u]$ is the range for the input values of each user):

$$\begin{aligned} &Masking.UnMask(k_{0,\tau}, \sum_{i=1}^{n} c_{i,\tau}) = \sum_{i=1}^{n} c_{i,\tau} - k_{0,\tau} \\ &= \sum_{i=1}^{n} (m_{i,\tau} + \sum_{j=1}^{i-1} k_{(i,j),\tau} - \sum_{j=i+1}^{n} k_{(i,j),\tau} - k_{(i,0),\tau}) - k_{0,\tau} \\ &= \sum_{i=1}^{n} m_{i,\tau} + \sum_{i=1}^{n} (\sum_{j=1}^{i-1} k_{(i,j),\tau} - \sum_{j=i+1}^{n} k_{(i,j),\tau}) \quad (\to 0) \\ &\qquad - \sum_{i=1}^{n} k_{(i,0),\tau} - k_{0,\tau} \\ &= \sum_{i=1}^{n} m_{i,\tau} - \sum_{i=1}^{n} k_{(i,0),\tau} - (-\sum_{i=1}^{n} k_{(0,i),\tau}) = \sum_{1}^{n} m_{i,\tau} \end{aligned} \tag{2}$$

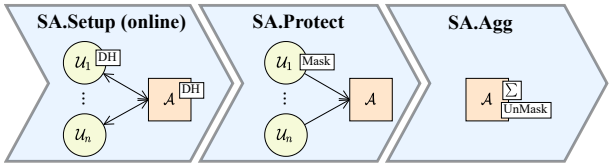

Figure 2: The main operations and the communication between parties in Masking-based SA.

Analysis: This scheme does not require a key dealer (KD) to distribute the masks. However, it relies on a trusted public key infrastructure (PKI). On the other hand, the masking operations are themselves very lightweight since they only include modular additions. However, the setup phase incurs significant overhead in terms of computation and communication costs per user which increase linearly with the total number of users. Since masking uses one-time pad encryption, the setup phase is performed on each

time period τ (notice the use of the tag τ for each key). Another disadvantage is that once keys are distributed, all users should provide their protected inputs (i.e., does not support dynamic users). Indeed, if some users did not participate, the masks on the aggregated value cannot be removed. Note that masking itself is information theoretically secure but the setup relies on a key agreement protocol that is computationally secure.

3.2.2 ***Secure Aggregation using AHE.*** A special type of Additively Homomorphic Encryption (AHE) schemes can be used for secure aggregation. Specifically, multi-user AHE are proposed such that "addition homomorphism" property is maintained across ciphertext generated by different users with different keys. These schemes are generally defined by the three following algorithms:

- $(pp, \{k_i\}_{i\in[1,..,n]}, k_0) \leftarrow AHE.Setup(\lambda)$: Given a security parameter λ, it generates the public parameters, the encryption keys, and the decryption key.
- $c_{i,\tau} \leftarrow AHE.Enc(pp, k_i, \tau, m_{i,\tau})$: It encrypts a message $m_{i,\tau}$ for a time period τ using the key k_i and outputs the ciphertext $c_{i,\tau}$.
- $\sum_1^n m_{i,\tau} \leftarrow AHE.Agg(pp, k_0, \tau, \{c_{i,\tau}\}_{i\in[1,..,n]})$: It evaluates the homomorphic operation on the n ciphertexts generated at the time period τ. Then decrypts the resulting ciphertext using the decryption key k_0.

A multi-user AHE scheme can guarantee *Aggregator Obliviousness* if each user encrypts only one input per time period. Several instantiations are proposed in [SCR+11, ET12, JL13, BJL16]. We present examples of AHE schemes that guarantee *Aggregator Obliviousness* in Appendix A.

Multi-user AHE schemes are designed specifically for secure aggregation protocols. So the SA phases are described as follows: in the **SA**.**Setup** phase, $\mathcal{TP}$ runs $AHE.Setup(\lambda)$ and distributes the keys to the users and the aggregator. The **SA**.**Setup** phase is executed only once; In the **SA**.**Protect** phase, $\mathcal{U}_i$ executes $AHE.Enc(pp, k_i, \tau, m_{i,\tau})$ and sends the ciphertext to the aggregator; Finally, in the **SA**.**Agg** phase, the aggregator executes $AHE.Agg(pp, k_0, \tau, \{c_{i,\tau}\}_{i\in[1,..,n]})$ and retrieves the sum of the inputs.

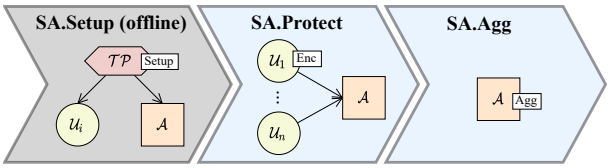

Figure 3: The main operations and the communication between parties in AHE-based SA.

Analysis: The main advantage of AHE schemes is that they require to run the setup phase only one time, and hence they are effective when aggregating a stream of data. This originally comes with a cost of relying on a trusted key dealer (KD) to perform the setup. Nevertheless, previous work has improved these schemes to enable running them without the need for a key dealer [LEM14b]. In terms of the computational cost per user, **SA**.**Protect** does not depend on the total number of users but incurs heavy operations. Similarly the communication cost per user does not depend on the total number of users but incurs a size expansion because of the size of the ciphertext. Additionally, similar to masking schemes, AHE does not support dynamic users since all users should provide their inputs to correctly aggregate them.

3.2.3 ***Secure Aggregation using Functional Encryption.*** Functional encryption (FE) is a type of encryption schemes that enables a user to learn a function on the encrypted data [BSW11]. Multi-Input Function Encryption (MIFE), introduced by Goldwasser [GGG+14], enables the learning of a function over multiple encrypted inputs. A special type of MIFE schemes can be designed to compute the inner product function of multiple inputs [AGRW17, ACF+18, DOT18]. Assuming we have two vectors x and y each consisting of l elements, the inner product of x and y is as follows:

$$IP(x, y) = \sum_{i=1}^{l} x[i]y[i] \tag{3}$$

An inner product MIFE scheme is defined by four algorithms:

- $\{pp, msk, \{k_i\}_{i\in[1,..,n]}\} \leftarrow MIFE.Setup(\lambda)$: Given a security parameter λ, it generates the public parameters pp, a master secret key msk, and n user keys $\{k_i\}_{i\in[1,..,n]}$.
- $c_{i,\tau} \leftarrow MIFE.Enc(pp, k_i, m_{i,\tau})$: It encrypts a message $m_{i,\tau}$ using the key k_i and outputs the ciphertext $c_{i,\tau}$.
- $dk_\tau \leftarrow MIFE.DKGen(pp, msk, y_\tau)$: It generates a decryption key dk_τ using the master secret key and a vector y_τ of n elements.
- $IP(m_\tau, y_\tau) \leftarrow MIFE.Dec(pp, dk_\tau, c_\tau, y_\tau)$: It takes the vector $c_\tau = [c_{1,\tau}, .., c_{n,\tau}]$, the vector y_τ, and the decryption key dk_τ generated from y_τ. It decrypts c_τ such that the result is the inner product of $m_\tau = [m_{1,\tau}, .., m_{n,\tau}]$ and y_τ.

MIFE schemes for inner product can be used to construct a secure aggregation protocol [XBZ+19, WPX+20]. In the **SA**.**Setup** phase, $\mathcal{TP}$ runs $MIFE.Setup(\lambda)$ and distributes the keys to the users. In the **SA**.**Protect** phase, $\mathcal{U}_i$ executes $MIFE.Enc(pp, k_i, m_{i,\tau})$ and sends the ciphertext $c_{i,\tau}$ to the aggregator. Finally, in the **SA**.**Agg** phase, the aggregator first sends a vector $y_\tau = [1, ..., 1]$ to $\mathcal{TP}$ which executes $MIFE.DKGen(pp, msk, y_\tau)$ and sends the decryption key dk_τ of time period τ to the aggregator. The aggregator then executes $MIFE.Dec(pp, dk_\tau, [c_{1,\tau}, .., c_{n,\tau}], y_\tau)$ and retrieve the inner product $\sum_{i=1}^n m_{i,\tau}y[i] = \sum_1^n m_{i,\tau}$.

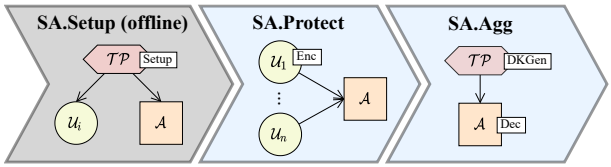

Figure 4: The main operations and the communication between parties in FE-based SA.

Analysis: Similar to AHE schemes, MIFE-based SA incurs constant computation and communication cost per user with respect to the total number of users. A very important property of these schemes is that it can deal with dynamic users by replacing zero weights in

the vector y_τ for the users that do not provide input at time period τ. On the other hand, the disadvantage of these schemes is that they require an online key dealer (KD) as a trusted third party to generate the decryption key for each time period.

3.3 MPC-based SA

Another cryptographic tool used to build secure aggregation protocols is multi-party computation (MPC). In MPC, keys are not needed to protect the user inputs. Instead, private messages are split into shares and distributed to multiple servers such that t of them can collaborate to reconstruct the private message. The schemes are also called t-out-of-n secret sharing and they are composed of two main algorithms:

- $\{[s]_i\}_{i\in[1,..,n]} \leftarrow MPC.Share(s,t,n)$: It splits a secret message s into n shares such that the secret can be reconstructed with t of the shares.
- $s \leftarrow MPC.Recon(\{[s]_i\}_{i\in U\subset[1,..,n]})$: It reconstructs the secret s from a subset of more than t shares ($\{[s]_i\}_{i\in U\subset[1,..,n]}$ where $|U| \geq t$).

A widely used MPC scheme is Shamir's secret sharing [Sha79]. This t-out-of-n sharing scheme uses polynomials to generate shares of secret values. Alternatively, a simpler n-out-of-n secret sharing can be constructed by simply splitting a secret value to n random values that sum up to the secret.

To design a secure aggregation protocol from MPC, the **SA.Setup** phase is not needed since no keys are generated. In the **SA.Protect** phase, a user protects its input by splitting it into l random shares using $MPC.Share(m_{i,\tau}, t, l)$ where l is the number of aggregators. It then sends one unique share to each aggregator. In the **SA.Agg** phase, each aggregator locally sums up the shares. Because secret sharing is additively homomorphic, the sum of the shares will result in a share of the sum. Finally, at least t aggregators broadcast their shares of the sum so that any aggregator can then run $MPC.Recon$ and retrieve the sum $\sum_1^n m_{i,\tau}$.

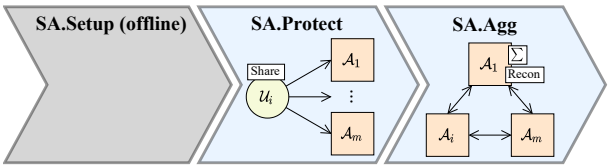

Figure 5: The main operations and the communication between parties in MPC-based SA.

Analysis: An important property of MPC-based SA is that it does not need a trusted third party since it does not need a key setup phase. Also, MPC supports dynamic users since it allows any subset of users to participate in the aggregation. This is mainly because MPC does not rely on secret keys that uniquely identifies a user. In the contrary, MPC incurs high computation and communication costs since the protection of a user input involves creating $O(nm)$ shares where n is the number of users and m is the size of the input. Furthermore, to distribute the shares, pre-existing secure channels are needed between the users and the aggregators. The secure channels ensure that each share is received and accessed only by its destined aggregator.

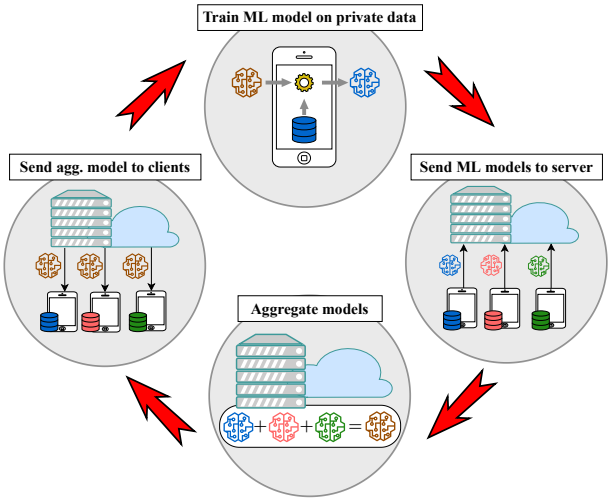

Figure 6: One federated learning round with 3 FL clients and the server.

4 SECURE AGGREGATION FOR FEDERATED LEARNING

Secure aggregation is used in federated learning to preserve the privacy of the clients. In this section, we first elaborate on the different categories of federated learning systems and we present the problem of inference attacks. Then, we show how secure aggregation based on cryptographic schemes can mitigate these attacks. We further study the suitability of the baseline SA categories presented in Section 3 to federated learning. To study their suitability, we identify the unique requirements of federated learning and we analyze whether the basic schemes presented in Section 3 meet these requirements. Finally, we survey the existing work on improving the existing SA protocols to cope with FL challenges, specifically. We propose to regroup these solutions based on the specific challenges they tackle.

4.1 Federated Learning and Inference Attacks

In FL, a model $\mathcal{M}$ is trained on n datasets $(\mathcal{D}_1, \mathcal{D}_2, ..., \mathcal{D}_n)$ each maintained by a different FL client. For each FL round τ, the client i receives the model $\mathcal{M}_\tau$ from the server and trains it on $\mathcal{D}_i$ which results in the trained model $\mathcal{M}_i^\tau$. The server aggregates the model and sends the aggregated model $\mathcal{M}^{\tau+1}$ back to the clients. Figure 6 illustrates one FL round.

Scale of federation. Based on the scale of the FL protocol, two types of FL can be distinguished: cross-silo FL and cross-device FL [KMA+19]. In cross-silo scenarios, a small number (order of tens) of powerful users host the data who often have decent computational power with a reliable and high bandwidth network connections. On the contrary, cross-device scenarios involve a large number of users. These users often correspond to end-devices with moderate to low computational power. In many applications, these devices interact directly with end-users from which they collect data.

Partitioning of the training data. There exist three categories of data partitioning [YLCT19, LWH19]: Horizontal partitioning, vertical partitioning, and a hybrid partitioning. In a horizontal partitioned dataset, each FL client holds a set of complete training samples. Each sample contains all the training features and the corresponding label. Hence, each client is able to train a local model on these samples. Differently, in a vertically partitioned dataset [DP21], a client may hold part of the features of each training sample while the other parts might be held by other FL clients. In this FL type, the clients are not able to locally train a model without collecting the missing information of each sample from other clients. A hybrid partitioned dataset is a combination of horizontally and vertically partitioned datasets. SA is only suitable to FL based on horizontally partitioned datasets since those based vertically partitioned datasets require more operations than just summing the clients' updates. Therefore, in the remaining of this section, FL for vertical and hybrid datasets will be omitted.

Learning algorithm. The most used learning algorithm for horizontal FL is *Federated Averaging* [MMR+17], which is based on Stochastic Gradient Descent (SGD) [iA93]. SGD is an iterative algorithm used to train a model on a dataset (i.e., find the best weights of a model that can fit the dataset). On each SGD step, the client i uses the current weights w^τ of the model $\mathcal{M}^\tau$ and a loss function $\mathcal{L}_f$ to compute the gradient g_i^τ from the values in its dataset $\mathcal{D}_i$.

$$g_i^\tau = \Delta\mathcal{L}_f(w^\tau, \mathcal{D}_i) = \Delta \sum_{(x,y)\in\mathcal{D}_i} \mathcal{L}_f(w^\tau, x, y)$$

Then, the gradient is used to update the weights of the model with a learning rate η ($w_i^\tau = w^\tau - \eta g_i^\tau$). The FL clients send their new trained model $\mathcal{M}_i^\tau$ represented by the computed weights w_i^τ to the FL server which aggregates them:

$$m_{i,\tau} \leftarrow w_i^\tau, \quad w^{\tau+1} \leftarrow \frac{\sum_1^n m_{i,\tau}}{n}$$

Finally, each FL client obtains the aggregated model $\mathcal{M}^{\tau+1}$ represented by the aggregated weights $w_i^{\tau+1}$ and starts a new federated learning round.

Inference Attacks. An adversary having access to the model updates sent by the clients can perform inference attacks. In more details, the attacker learns some private information about the clients' datasets. These attacks could consist of membership inference attacks [SSSS17a, NSH19] where the attackers learn whether a specific data record is part of the training dataset or not. One can also consider data reconstruction attacks [DN03, WLW+09] (a.k.a. model invasion attacks [FJR15]) where the attacker learns some of the attributes of a record in the dataset. Furthermore, Ateniese et al. [AMS+15] and Ganju et al. [GWY+18] present data properties inference attacks where the attacker learns global properties of the training dataset, such as the environment in which the data was produced.

4.2 SA to Mitigate Inference Attacks on Federated Learning

Secure aggregation based on cryptographic schemes aims to prevent inference attacks by hiding the model updates from any potential adversary. Based on the definition given in Section 3, it involves

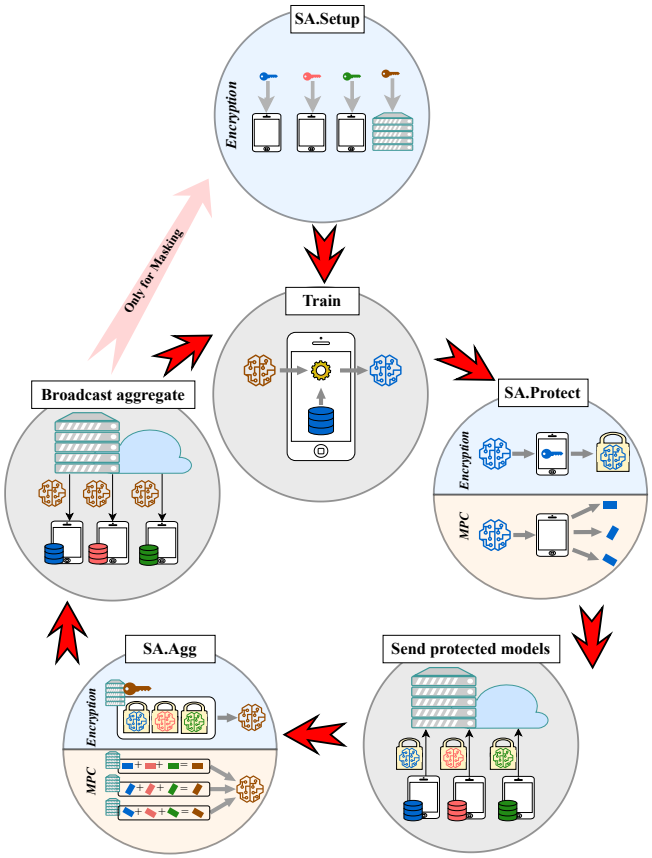

Figure 7: A secure aggregation protocol integrated in federated learning. The secure aggregation protocol ensures that the aggregators do not learn anything about the clients' locally trained ML models except their aggregate.

two main players (i.e., users $\mathcal{U}$ and aggregators $\mathcal{A}$) which execute the three SA phases (i.e., **SA**.**Setup**, **SA**.**Protect**, and **SA**.**Agg**). The users correspond to the FL clients and their inputs in each round are the locally trained models ($m_{i,\tau} \leftarrow w_i^\tau$). On the other hand, the aggregator (or the set of aggregators) act as the FL server. Any secure aggregation algorithm consisting of the three defined phases can be used for running a secure version of the FL protocol. To run FL with secure aggregation, **SA**.**Setup** phase is performed before the training starts. Then for each FL round τ, the client $\mathcal{U}_i$ trains its model on its local data and obtains the gradient w_i^τ. It then runs **SA**.**Protect** to protect the local trained model and send it to the server. Finally, the server runs **SA**.**Agg** after it collects all protected trained models. As a result, the clients get the aggregated model and starts a new FL round. Figure 7 shows the components of secure aggregation integrated in federated learning.

4.3 The Challenges in using SA for FL

Nowadays, the use of SA based on cryptographic schemes in federated learning becomes increasingly popular. We already witness several federated learning frameworks such as FATE [Dep21], Paddle FL [Pad21], and Pysyft [Ope21] integrating these technologies. Nevertheless, these implementations are not practical in real-world

	Masking-based SA	AHE-based SA	FE-based SA	MPC-based SA
Client Failure ($\mathbb{C}_1$)	○	○	●	●
High Dim. Inputs ($\mathbb{C}_2$)	◐	◐	●	○
Scalability ($\mathbb{C}_3$)	○	●	●	○
Privacy Leaks ($\mathbb{C}_4$)	○	○	○	○
Malicious Users ($\mathbb{C}_5$)	○	○	○	○
Malicious Agg ($\mathbb{C}_6$)	○	○	○	○

Table 2: The challenges of using SA for FL based on the unique requirements of FL. It shows for each challenge wether the baseline SA protocols defined in Section 3 can originally cope with that challenge.

scenarios since they underestimate the impact of SA schemes based on cryptographic schemes on FL. Indeed, federated learning features some unique properties and characteristics that differ from previous applications where SA was used. We hereby identify six unique properties for FL that raise significant challenges for the integration of SA in FL. We further analyze the suitability of each SA category (see Section 3) to cope with these characteristics. We summarize the results in Table 2.

4.3.1 Failures and drops of clients at realtime ($\mathbb{C}_1$). In cross-device FL scenarios, it is common to have mobile, unreliable FL clients. The mobility of a client may cause failures (drops) of some FL clients causing their unavailability for some federated learning rounds. Failures of clients may even happen within the FL round as well. All this can be a problem for some secure aggregation schemes that do not support dynamic users. In particular, SA schemes based on masking and AHE are not fundamentally designed to cope with user failures. Therefore, the need for fault-tolerant secure aggregation is a new requirement for FL.

4.3.2 Client's inputs are vectors of high dimension rather than integers ($\mathbb{C}_2$). In FL, the user's input is a vector that holds all the model parameters (weights). Not all types of secure aggregation protocols can work efficiently with vectors. For example, MPC-based SA incurs a significant communication cost since shares of the inputs have the same size of the input. Therefore, it is not practical to run secret sharing to share large vectors. Also, in masking-based SA, the masks have the size of the input; Hence, agreeing on the masks should be efficient in terms of bandwidth consumption. Additionally, for AHE, it is not practical to encrypt each element of the input vector. This calls for efficient packing techniques designed for AHE-based SA.

4.3.3 Huge number of clients ($\mathbb{C}_3$). Recently, we start to observe FL applications involving thousands of FL clients. Google is researching how to train Gboard (the Android's keyboard application) search suggestion system using federated learning on large scale [YAE+18, HKR+18]. With secure aggregation integrated with federated learning, the scalability problem becomes a serious challenge. MPC-based SA protocols do not scale well for huge number of users since they suffer from a quadratic complexity in terms of communication and computation. Similarly, masking-based SA suffers from a quadratic complexity in the setup phase. Additionally, with large number of clients, the typical synchronized FL protocol is not practical. In an asynchronous FL protocol, clients do not wait for the updates of a sufficient number of users at each FL round. Instead, the updates of the users are incorporated as soon as they arrive at the server. Adopting SA for asynchronous FL is challenging because updates may be protected with keys corresponding to different FL rounds.

4.3.4 Privacy attacks that bypass SA ($\mathbb{C}_4$). The aggregated model $M^{\tau+1}$ is a public information that is accessible for all FL clients. Therefore, secure aggregation is not used to hide this value. There exist a different type of inference attacks that can still infer private information from the aggregated model, only [SSSS17b]. For example, recently So et al. [SAG+21] pointed out a new attack to leak the client's updates even when protected with secure aggregation. The authors notice that the models from the FL clients do not change a lot between one training step and another one when the trained model starts to converge. This causes a privacy leakage if a FL client did not participate. In detail, if all FL clients participate in round $\tau - 1$ and all clients except one participate in round τ, and if the inputs did not change a lot, an adversary who has access to the aggregated model updates for rounds τ and $\tau - 1$ will be able to approximate the inputs of the missing FL client. Such specific attacks can bypass the security measures of SA. Gao et al. [GHG+21] implemented these types of attacked and showed how they can effectively infer the category of given data samples.

Secure aggregation protocols by definition do not provide protection against these types of attacks. Therefore, additional security mechanisms should be used with secure aggregation to mitigate these attacks.

4.3.5 Malicious Users ($\mathbb{C}_5$). Earlier SA protocols proposed before the appear of the FL paradigm appeared, consider a *honest-but-curious* threat model with *colluding* users (see Section 2). Such threat model is not sufficient in the context of federated learning. Specifically, FL clients cannot be trusted to provide their inputs truthfully at each FL round. Thus, we should consider an extended threat model which considers malicious users.

Indeed, poisoning attacks (a.k.a., backdooring attacks) are attacks where malicious FL clients manipulate their model updates M_i^τ to affect the aggregated model $M^{\tau+1}$. Their goal is to install a backdoor in the trained model. A "backdoored" model behaves almost normally on all inputs except for attacker-chosen inputs at which it outputs attacker-desired predictions. Malicious FL clients use two main methods to poison a model: Dataset poisoning [STS16] where attackers insert malicious records in their dataset; and model poisoning [BVH+20a] (a.k.a., constrain-and-scale attacks) where the attacker replaces the trained model by a malicious model and send it instead of the trained model. An even more recent attack method is distributed poisoning attacks [XHCL20] in which the poison is distributed among several malicious clients inputs so it is harder to detect malicious models. On the other hand, malicious clients can perform less stealthy attacks by sending ill-formed inputs to prevent the calculation of the aggregation.

All SA protocols studied in Section 3 are designed to achieve *Aggregator Obliviousness* in the *honest-but-curious* model. To further prevent poisoning attacks we need additional security mechanisms for SA.

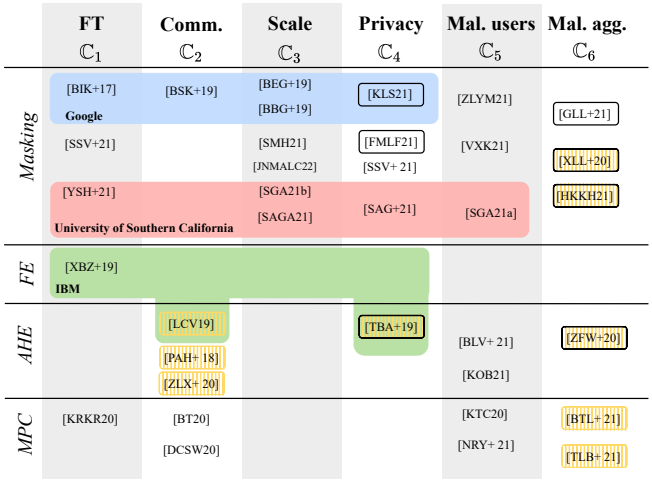

Figure 8: A summary of the existing FL solutions that use crypto-based secure aggregation grouped by the type of the SA used and the specific challenge they tackle. Bordered boxes indicates that the solution presents a technique that can be deployed in other types of SA protocols (eg., [XLL+20] is implemented on masking-based SA but can be used also for AHE-based SA). Hatched boxes indicates that the scheme used cannot achieve the security requirements since they do not support collusions (this is discussed in Section 5). Colors represents research groups of the authors.

4.3.6 Malicious Aggregator ($\mathbb{C}_6$). Similar to challenge ($\mathbb{C}_5$), the *honest-but-curious* threat model is not sufficient to prevent the cheating of a server in the context of federated learning. More specifically, the SA protocols described in Section 3 prevent a curious FL server from learning the clients inputs, but cannot protect against a malicious server that modifies the aggregated model. Indeed a malicious server can cause a huge damage because it has full control of the final aggregated value. Therefore, an adversary controlling the FL server can force the clients to learn an adversary chosen model. In fact, the impact of a malicious aggregator can even go beyond forging the aggregation result. Pasquini et al. [PFA21] showed that a malicious aggregator can even compromise the privacy by bypassing the secure aggregation protocol. An example for these attacks illustrated by the authors is when the malicious aggregator chooses specific values for the aggregated result. The values are chosen such that when the clients train the forged model sent by the aggregator, the training outputs a model of zero parameters. Hence, the malicious aggregator can suppress arbitrary clients of his choice from the aggregation by sending them malformed models. Therefore, it can suppress all clients except a targeted one and leak its input.

To prevent such attacks, SA protocols should consider a malicious aggregator in their threat model.

4.4 Crypto-based SA solutions designed for FL

Many research has been performed on designing SA protocols based on cryptographic schemes for federated learning applications. Most of the proposed schemes are improvements of the basic secure aggregations protocols described in Section 3 and tackle one or more of the aforementioned challenges ($\mathbb{C}_1$-$\mathbb{C}_6$). The proposed schemes can be categorized based on the challenge they tackle. We summarize how these solutions propose different solutions for each of the challenges. Table 3 presents an overview of these solutions grouped by their challenge scope. Also, Figure 8 regroups them per SA category and shows the relation between the solutions.

4.4.1 Fault-tolerant SA. To tackle the problem of client failures (see $\mathbb{C}_1$), a fault-tolerant secure aggregation protocol should be used. MPC-based secure aggregation, specifically, Shamir's SS scheme [Sha79] is by design fault-tolerant. It is used in [DCSW20, KRKR20] where the FL server is replaced with a set of separated aggregators. The high communication cost these solutions incur encouraged researchers to look for alternative fault-tolerant solutions.

MIFE-based schemes are also by design fault-tolerant since the data aggregator can assign zero weights for missing clients [XBZ+19]. However, these schemes require a key-dealer to stay online for each federated learning round.

On the other hand, Bonawitz et al. [BIK+17] proposed a fault-tolerant variant of the masking-based SA. Later, this scheme was widely adopted and improved by [BBG+20, EA20, SGA21b, KLS21, XLL+20, GLL+21]. The idea of this scheme is to merge Shamir's SS scheme with masking. This merging can achieve best of both worlds. Specifically, it benefits from the lightweight operations and low communication overhead of the masking scheme and on the other hand, it benefits from the fault-tolerance property of Shamir's SS scheme. Thanks to this tradeoff, it is considered as a big jump towards designing practical secure aggregation scheme for cross-device FL scenarios. The details of this scheme are explained in Appendix C. A similar scheme is proposed by Stevens et al. [SSV+21] that replaces the standard masking with a Learning With Error masking and used a packed and verifiable version of Shamir's secret sharing. Also, Yang et al. proposed LightSecAgg [YSH+21] which replaces the Shamir's secret sharing scheme with a secret sharing scheme based on Maximal Distance Seperable (MDS) code [RL89]. The work of Yang et al. reduces the computation time on the server. Another approach is proposed by Swanand et al. [KRKR20] that uses Fast Fourier Transform (FFT) for secret sharing.

4.4.2 Communication efficient SA. Researchers propose some techniques to bound the overhead incurred by SA (see $\mathbb{C}_2$). For encryption based schemes, batch encryption has been leveraged by Liu et al. [LCV19], Phong et al. [PAH+18], and Yang et al. [ZLX+20]. In BatchCrypt [ZLX+20], the authors propose a method to quantize and batch the elements of a gradient before encryption. The strength of their approach is that it preserves the additively homomorphic property of the ciphertexts. Alternatively, Beguier et al. [BT20] propose a compression technique to decrease the size of the inputs and then use secret sharing over the compressed results. Another interesting technique presented by Wu et al. [WPX+20] is to use All Or Nothing Transformation (AONT) [Riv97]. The authors show that by transforming the client's gradient with AONT, it is sufficient to encrypt a small part of the transformed gradient. This can decrease the size of the protected user input by several orders. Another method followed by Elkordy et al. [EA20] and Benawitz et al. [BSK+19] is to use auto-tuned quantization. This technique is

Scope	Solution (year)	SA scheme	FL scale	Technique
Fault-Tolerance ($\mathbb{C}_1$)	Bonawitz et al. [BIK+17]	Masking	X-Dev	Integrating Shamir SS with Masking
	HybridAlpha [XBZ+19]	MIFE	X-Silo	Assigning zero weights for dropped clients
	FastSecAgg [KRKR20]	MPC-t-of-n	X-Dev	Using FFT with Shamir SS
	Stevens et al. [SSV+21]	Masking	X-Dev	Integrating Shamir SS with LWE-based Masking
	LightSecAgg [YSH+21]	Masking	X-Dev	Integrating MDS code with Masking
Communication Efficiency ($\mathbb{C}_2$)	Phong et al. [PAH+18]	AHE	X-Silo	Batch encryption
	Bonawitz et al. [BSK+19]	Masking	X-Dev	Autotuned quantization
	Liu et al. [LCV19]	AHE	X-Silo	Batch encryption
	BatchCrypt [ZLX+20]	AHE	X-Silo	Batch encryption
	Wu et al. [WPX+20]	MIFE	X-Silo	All or nothing transformation
	Safer [BT20]	MPC-n-of-n	X-Silo	Compression
	HeteroSAg [EA20]	Masking	X-Dev	Autotuned quantization
	EastFly [DCSW20]	MPC-n-of-n / AHE	X-Silo	Quantization: ternary FL
Scalability ($\mathbb{C}_3$)	Bonawtiz et al. [BEG+19]	Masking	X-Dev	Sub-grouping: Running multiple SA instances
	Bell et al. [BBG+20]	Masking	X-Dev	Sub-grouping: Creating random connected graphs
	TurboAgg [SGA21b]	Masking	X-Dev	Sub-grouping: Circular subgroups of clients
	SAFE [SMH21]	Masking	X-Dev	Arranging clients in a circular chain
	So et al. [SAGA21]	Masking	X-Dev	Adapting SA for asynchronous FL
	SwiftAgg [JNMALC22]	Masking	X-Dev	Sub-grouping: Running multiple SA instances
Privacy Enhancing ($\mathbb{C}_4$)	Truex et al. [TBA+19]	AHE	X-Silo	DDP: clients add gaussian noise
	Peter et al. [KLS21]	Masking	X-Dev	DDP: clients add gaussian noise
	So et al. [SAG+21]	Masking	X-Dev	Multi-round privacy using client selection
	Timothy et al. [SSV+21]	Masking	X-Dev	DDP: clients use LWE-based masking
	Joaquín et al. [FMLF21]	Masking	X-Dev	DP: aggregator add noise to the aggregate
Verify Inputs ($\mathbb{C}_5$)	MLGuard [KTC20]	MPC-2-of-2	X-Silo	Boolean circuits to compute cosine distance
	FLGuard [NRY+21]	MPC-2-of-2	X-Silo	Bool/Arth circuits to perform clustering
	RoFL [BLV+21]	AHE	X-Silo	Commitment scheme to compute euclidean distance
	BREA [SGA21a]	Masking	X-Dev	Arithmetic circuits to compute square distance
	Karakoc et al. [KOB21]	AHE	X-Silo	OPPRF to compare with a threshold
	SAFELearning [ZLYM21]	Masking	X-Dev	Multi-step aggregation to verify intermediate results
	Velicheti et al. [VXK21]	Masking	X-Dev	Multi-step aggregation to verify intermediate results
Verify Agg. ($\mathbb{C}_6$)	Zhang et al. [ZFW+20]	AHE	X-Silo	Using HHF based on Bilinear Maps
	VerifyNet [XLL+20]	Masking	X-Dev	Using HHF with ZKP scheme
	VERSA [HKKH21]	Masking	X-Dev	Using keyed HHF with ZKP scheme
	NIVA [BTL+21]	MPC	X-Dev	Verifiable secret sharing
	DEVA [TLB+21]	MPC	X-Dev	Verifiable secret sharing
	VeriFL [GLL+21]	Masking	X-Dev	Using commitment scheme with HHF

Table 3: Categorization of secure federated learning solutions based on the challenge tackled with a short description of the proposed solution. All the solutions are secure in the HBC model except those addressing $\mathbb{C}_6$ (malicious aggregator) and $\mathbb{C}_5$ (malicious users) thus addressing a specific malicious setting. An exception is for the solutions in red which don't protect against collusions between users and aggregators and thus are considered not secure (based on our security definitions in Section 2).

integrated with FT-Masking [BIK+17] and enables the data owners to adapt their quantization level based on the requirements. One very important aspect for these quantization techniques is their impact on performance. Therefore, all these mentioned work study the trade-off between reducing the size of the protected gradient and the maintaining a high precision for the trained model.

4.4.3 Scalable SA. To tackle challenge $\mathbb{C}_3$, scalability of SA started to gain researchers' attention thanks to the new large-scale applications of FL. Bonawitz et al. [BEG+19] set up a general framework to scale a secure aggregation framework to millions of devices. The authors propose to simply run multiple instances of the scheme, one for each subgroup of clients. Each subgroup compute intermediate aggregates which are combined later. The same intuition of grouping clients is followed up by Bell et al. [BBG+20] and by So et al. [SGA21b]. Bell et al. observe that the FT-Masking scheme in [BIK+17] does not require that all the clients need to be connected. Thus, they propose to generalize the scheme by creating random graphs. Each FL client executes the FT-Masking with its neighbors. The new protocol assumes that not all the neighbors will be corrupted at the same time and it proposes a method to build the so called "good" graphs. Similarly, both So et al. (TurboAgg) [SGA21b] and Sandholm et al. (SAFE) [SMH21] propose a circular topology. Clients perform a chain of aggregations by passing the

aggregated updates to the next client. To further deal with large number of client, So et al. [SAGA21] propose a SA protocol that can be integrated in asynchronous FL. The solution uses the scheme proposed in [YSH+21] and adapts it to enable securely aggregating inputs from different time periods.

4.4.4 SA resilient to privacy attack. To deal with inference attacks on the aggregated model ($\mathbb{C}_4$). Differential Privacy (DP) [Dwo06] should be used with secure aggregation. Notice that using DP with SA is to protect the aggregated model, only and not user inputs.

A simple method is to let the aggregator apply DP on the aggregated model [FMLF21]. However, this requires to trust the aggregator. A better method is to use a distributed version of DP (DDP) along with SA to mitigate the information leakage caused by the public aggregated model. Few works have followed this approach for FL [KLS21, TBA+19]. These solutions add Gaussian noise to the FL clients' inputs. They leverage the fact that FL clients inputs are protected with cryptographic tools (thanks to SA) which permit them to decrease the level of noise while achieving sufficient privacy level. Therefore, using DDP with SA limits the degradation of the accuracy of the trained model compared to using DP alone. Stevens et al. [SSV+21] followed a similar approach by using Learning with Error (LWE) masking technique to make the final aggregate differentially private.

On the other hand, a multi-round privacy concept is introduced by So et al. [SAG+21]. This concept is to ensure that an adversary cannot learn valuable information by monitoring the changes in the aggregated model across different FL rounds. The authors propose a solution enlightened by the work done by [TNW+21]. They propose to randomly and fairly (using weights) select participants in each FL round based on well-defined criteria so called *Batch Partitioning*. Using this technique they can guarantee the long-term privacy of the data at the FL clients.

4.4.5 SA against malicious users. To deal with malicious users that perform *poisoning attacks* ($\mathbb{C}_5$), the FL server needs a mechanism to validate the inputs of the clients. Mitigating poisoning attacks is studied by researchers independently from using secure aggregation for FL [FYB18, AMMK20]. One of the methods used to prevent such attacks is to use the cosine distance [FYB18] to detect poisoned inputs that deviate from the other benign inputs. Clustering [STS16, BEMGS17] and anomaly detection methods [AMMK20] are also used to detect malicious model updates. An orthogonal approach is to use clipping and noising to smooth the model updates and remove the differences [BVH+20b]. While all these solutions are shown to be efficient in preventing poisoning attacks, using them with secure aggregation is a big challenge. The problem is that all these solutions rely on analyzing the FL clients' inputs while secure aggregation aims to hide and protect these inputs.

Several methods are proposed to verify the inputs while keeping them protected to preserve their privacy. For MPC-based SA, it is possible to build circuits that can perform complex operations on the shares. This can be used to evaluate functions on the inputs other than just computing the sum. Indeed, MLGuard [KTC20] proposes to verify the users' inputs by transforming a verification function into a circuit which gets executed by the two servers using 2PC. The verification function computes the distance between the clients' inputs. The circuit compares the distance to pre-defined thresholds and thus reject the input. FLGuard [NRY+21] follows the same approach by building two circuits. One circuit for detecting poisoned inputs using a dynamic clustering algorithm (HDBSCAN [CMS13]) and another circuit for reducing the impact of poisoned inputs using clipping and noising. The communication cost of running these circuits is significant thus making scalability even harder to achieve for SA in the federated learning context. A promising approach to reduce this cost is through the use of *secret-sharing non-interactive proof (SNIP)*. This approach was proposed in [CGB17] (Prio). Using SNIP enables the aggregators to validate the user inputs without interacting with the users and with minimal interaction between themselves. This scheme is not yet deployed in FL applications. *SNIP* brings a great advantage over standard 2-PC validation circuits since it does not limit the number of aggregators thanks to its lower communication cost. The limitation of SNIP is that it only supports specific validation functions. Therefore, it is an open challenge to design validation circuits for detecting poisoning attack using SNIP.

On the other hand, regarding AHE-based SA, Karakoc et al. [KOB21] propose *OPPRF*, an algorithm based on private set membership (PSM) [CO18] and oblivious transfer (OT) [NP05]. *OPPRF* uses PSM to perform equality checks between values (i.e., equivalent to finding intersection between sets of cardinality equal to one [Cou18]). Using *OPPRF*, the users can create tags that are only valid if their inputs are lower than a threshold provided by the aggregator. Karakoc et al. [KOB21] applied this scheme for Multi-Key AHE secure aggregation schemes and evaluated it in SFL applications. The scheme enables the FL server to detect poisoning attacks by checking that the minimum, maximum and the average of the gradient elements does not cross a certain threshold value. The threshold is configured based on an observation of the gradients of benign clients. Another approach is proposed by Lukas et al. [BLV+21]. The authors use a *non-interactive commitment* scheme proposed in [Ped92]. Using this scheme, the users create proofs that the Euclidean distance of their inputs satisfies the bounds set by the aggregator. Upon receiving the client protected input and the commitment, the server verifies that the proof is valid.

For masking-based SA, two techniques are proposed. One technique proposed by So et al. [SGA21a] in which users secretly share their model updates with all other clients and then compute the square distance between the model shares. The server can finally reconstruct the square distances and use the result to detect malicious inputs. An alternative technique is proposed by Zhang et al. [ZLYM21] and Velicheti et al. [VXK21]. In details, users are anonymously and randomly grouped into clusters. Aggregation happens per cluster and then a second round of aggregation happens on the results of each cluster. For each cluster, the intermediate aggregation results are checked to prevent poisoning attacks. The fact that attackers do not know to which cluster the compromised device belongs to, protects from distributed poisoning attacks (see $\mathbb{C}_5$).

4.4.6 SA against malicious aggregator. In a malicious aggregator threat model, the FL server forges false aggregation results ($\mathbb{C}_6$). Mitigating these attacks requires a verifiable secure aggregation scheme. Many solutions are proposed to enable the verification of

the aggregation outcome [KShS12, SS11, DOS18, CDE+18]. However, these solutions do not fit well federated learning applications due to their high communication overhead. In the context of federated learning, six solutions are proposed [GLL+21, HKKH21, XLL+20, ZFW+20, TLB+21, BTL+21].

For masking-based and AHE-based SA, Zhang et al. [ZFW+20] and Xu et al. (VerifyNet) [XLL+20] used *Homomorphic Hash Functions (HHF)* to verify the result of the aggregation. *HHF* can be build using bilinear maps [BGLS03, Fre12]. First, the data owners create authentication tags of their inputs and send them to the aggregator. The latter aggregates them to prove the outcome of the aggregation. Finally, the aggregator verifies the result. Hahn et al. [HKKH21] detected possible brute-force attacks on VerifyNet and improved it by deploying a keyed *HHF*. All these solutions do not support collusions between the users and the aggregator. Therefore, cannot be considered secure based on our security definitions (see Section 2). Another problem with these solutions is that they significantly affect the performance of SA. This is because of the linear increase in computation and communication overhead with the increase of the dimension of inputs. This is a clear limitation since the performance of the ML model highly depends on their size (i.e., number of parameters). To solve this problem, Guo et al. (VeriFL) [GLL+21] focus on designing a verification scheme specifically for secure aggregation applications with inputs of high dimension. To support user-aggregator collusions, the authors integrated a *commitment* scheme to prevent users from changing their hashes after the computation of the aggregate. The authors of VeriFL apply this scheme to the fault-tolerant masking scheme in [BIK+17]. The evaluation on federated learning application show a significant reduction in communication overhead with respect to other verification schemes. However, VeriFL still suffers from a quadratic computation and communication cost with respect to the number of FL clients. Achieving a better scalability for verification systems is an open problem.

For MPC-based SA, Brunetta et al. [BTL+21] propose NIVA as a non-interactive secure aggregation protocol that includes the verification of the result. The users create a tag for each of their input shares. Upon computing the aggregate, the result can be verified using all the generated tags. Tsaloli et al. [TLB+21] proposes DEVA which improves the number of tags created for each user. DEVA requires that a user creates a single tag for its input rather than creating a tag per for each share. Both approaches do not support collusions between users and the aggregator and incur very high communication overhead since they use MPC.

5 TAKEAWAY MESSAGES AND OBSERVATIONS

We have extensively studied the federated learning solutions that integrate secure aggregation schemes. In this section, we identify and share the following observations and takeaway messages:

$\mathbb{O}_1$ We can clearly see that masking-based SA are the most integrated secure aggregation solution for federated learning. More specifically FT-Masking [BIK+17], appeared in 20 solutions where each one tried to improve it in a certain direction. It will be very nice to see all these parallel improvements integrated in a single solution.

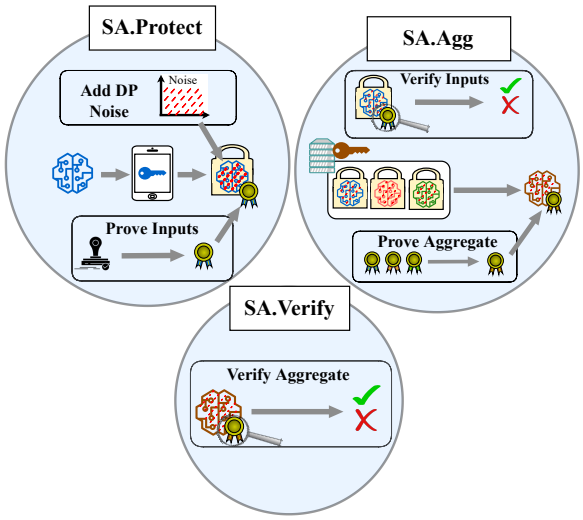

Figure 9: New Components of Secure Aggregation

$\mathbb{O}_2$ We notice that secure aggregation solutions based on AHE are not widely adopted in federated learning. This is mainly because they do not support user dynamics. However, we see that AHE-based SA is promising since they provide long-term security using the same user keys. We hope to see more research improving these schemes towards a practical deployment in federated learning context.

$\mathbb{O}_3$ We notice that some of the solutions proposed to preserve the privacy in federated learning do not adhere to the minimal security requirements for secure aggregation protocols (see *Aggregator Obliviousness* in Section 2). Specifically, AHE schemes [PAH+18, LCV19, ZLX+20, ZFW+20] and masking-based verification schemes [ZFW+20, XLL+20, HKKH21] that use the same key for all usersp should not be considered secure since they do not guarantee security in case of a collusion between a user and the aggregator.

$\mathbb{O}_4$ We note that secure aggregation alone is insufficient to guarantee the privacy of the clients datasets in the context of federated learning. Although SA helps prevent inference attacks, the global model that is collaboratively computed from private individual inputs can still leak information. Therefore, additional protection mechanisms are required. For this purpose, differentially private mechanism and multi-round privacy are suitable candidates to cope with this problem.

$\mathbb{O}_5$ Poisoning attacks against federated learning call for some integrity mechanisms that would allow the aggregator (the FL server in this context) verify the correctness/veracity of received inputs. Nevertheless, the cost of such mechanisms can be significant. Therefore, we can consider the design of such mechanisms that can (i) detect stealthy and sophisticated poisoning attacks, (ii) ensure the security and scalability requirements as an open challenge.

$\mathbb{O}_6$ Similar to the previous observation, we identify the need for an integrity mechanism for verifying the correctness of the actual aggregate. Basic solutions would linearly increase the size of the transmitted data between parties w.r.t. the model size. Using incremental HHF is promising as shown in VeriFL [GLL+21]. However, this solution is still far from being applied for FL application in larger scales since it still implies a linear increase in communication and computation cost w.r.t the number of clients.

Based on all the previous observations, we propose to revisit the definition of crypto-based secure aggregation to make it suitable for FL. Specifically, we revise the description of the protocol phases (i.e., **SA**.**Setup**, **SA**.**Protect**, and **SA**.**Agg**) to meet all the security requirements for FL application. Following observation $\mathbb{O}_4$, the **SA**.**Protect** phase should be modified such that users first pre-process their inputs with distributed differential privacy mechanism before running the actual protection algorithm. Additionally, based on observation $\mathbb{O}_5$, **SA**.**Protect** should also generate integrity proofs of inputs which are sent together with the protected inputs to the data aggregators. On the other hand, **SA**.**Agg** should include a verification mechanism of the inputs which validates the integrity proofs. Moreover, observation $\mathbb{O}_6$ indicates that **SA**.**Agg** should compute a proof of the aggregation which is sent to the users along with aggregation result. In order for the users to validate the aggregation result, we require an additional secure aggregation phase. Namely, **SA**.**Verify** phase which is performed as a final step by the users. In this phase, the users verify the received result of the aggregation.

In summary, we propose a better definition of secure aggregation protocols based on cryptographic schemes which copes with the security requirements of federated learning. The defintion consists of four phases: **SA**.**Setup**, **SA**.**Protect**, **SA**.**Agg**, and **SA**.**Verify**. Figure 9 shows the details of the improvements in each of the phases. It is worth to note that this new definition combines and generalizes all the improvements proposed by the state-of-the-art solutions. It would be interesting to develop the first SA solution for FL implementing our proposed definition by combining all the state-of-the-art techniques.

6 RELATED STUDIES

To the best of our knowledge, there are no systematization of knowledge in the literature focusing on secure aggregation ptotocols based on cryptographic schemes and their application to federated learning protocols. Recent works [KMA+19, SWRH20, MPP+21] studied federated learning and the security and privacy issues of machine learning. In all these three studies, the authors present a high level overview of all the possible secure aggregation techniques that can be deployed in federated learning. With respect to our work, we concentrate our research only on the solutions based on cryptographic schemes by performing an in-depth systematic study that results in a clear comparison of these solutions. For example, none of theses studies give a formal definition of this concept and shows how all the existing techniques can be instantiated under this definition. More importantly, we extensively study the use of such SA protocols for federated learning in the literature. Therefore, we identify the specific challenges of using these SA protocols in FL and we analyze the existing solutions.

In a recent work [CPTPH21], the study focuses on a cross-field systematization of knowledge on privacy-preserving collaborative training of tree-based models such as decision-trees, random forests, and boosting. The systematization was based on the learning algorithm, the collaborative model, the protection mechanism, and the threat model. In [HMSY21], the authors review and analyze techniques and protocols used for privacy-preserving clustering with respect to efficiency, privacy, and security models. The above studies focus on specific types of machine learning models that require special treatment to be trained collaboratively. In contrary, our study focuses on general machine learning models that can be trained using horizontal federated learning.

7 CONCLUSION

In this paper, we proposed a formal definition of secure aggregation based on cryptographic schemes and provided an overview of the literature. We have first categorized the solutions based on the underlying cryptographic technique and further studied them with respect to their architecture, performance and their support of dynamic users. We then focused on the use of secure aggregation in federated learning. We have identified six main challenges: FL clients' failures, large dimension of client's input, huge number of clients, inference attacks and multi-round leakage, poisoning attacks from malicious users, and forged aggregation from malicious aggregator. We studied the existing 37 SFL solutions and categorized them based on the challenge they tackle and the attack they mitigate. Thanks to this study, we were able to present six main observations that we hope will help point the research in this field to the right direction.

REFERENCES

[ÁC11] Gergely Ács and Claude Castelluccia. I have a dream! (differentially private smart metering). In *Information Hiding*. Springer Berlin Heidelberg, 2011.

[ACF+18] Michel Abdalla, Dario Catalano, Dario Fiore, Romain Gay, and Bogdan Ursu. Multi-input functional encryption for inner products: Function-hiding realizations and constructions without pairings. In *Advances in Cryptology – CRYPTO 2018*. Springer International Publishing, 2018.

[ADMC17] Muhammad Rizwan Asghar, György Dán, Daniele Miorandi, and Imrich Chlamtac. Smart meter data privacy: A survey. *IEEE Communications Surveys Tutorials*, 2017.

[AGRW17] Michel Abdalla, Romain Gay, Mariana Raykova, and Hoeteck Wee. Multi-input inner-product functional encryption from pairings. In *Advances in Cryptology – EUROCRYPT 2017*. Springer International Publishing, 2017.

[AMMK20] Sébastien Andreina, Giorgia Azzurra Marson, Helen Möllering, and Ghassan Karame. Baffle: Backdoor detection via feedback-based federated learning. *CoRR*, abs/2011.02167, 2020.

[AMS+15] Giuseppe Ateniese, Luigi V. Mancini, Angelo Spognardi, Antonio Villani, Domenico Vitali, and Giovanni Felici. Hacking smart machines with smarter ones: How to extract meaningful data from machine learning classifiers. *Int. J. Secur. Netw.*, 10(3), sep 2015.

[ASY+18] Naman Agarwal, Ananda Theertha Suresh, Felix Yu, Sanjiv Kumar, and H. Brendan McMahan. Cpsgd: Communication-efficient and differentially-private distributed sgd. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18. Curran Associates Inc., 2018.

[BBG+20] James Henry Bell, Kallista A. Bonawitz, Adrià Gascón, Tancrède Lepoint, and Mariana Raykova. Secure single-server aggregation with (poly)logarithmic overhead. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, CCS '20. Association for Computing Machinery, 2020.

[BEG+19] Kallista A. Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloé Kiddon, Jakub Konečný, Stefano Mazzocchi, H. Brendan McMahan, Timon Van Overveldt, David

Petrou, Daniel Ramage, and Jason Roselander. Towards federated learning at scale: System design. *CoRR*, abs/1902.01046, 2019.

[BEMGS17] Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. Machine learning with adversaries: Byzantine tolerant gradient descent. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17. Curran Associates Inc., 2017.

[BGLS03] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In *Advances in Cryptology — EUROCRYPT 2003*. Springer Berlin Heidelberg, 2003.

[BIK+17] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. CCS '17, pages 1175–1191, New York, NY, USA, 2017. Association for Computing Machinery.

[BJL16] Fabrice Benhamouda, Marc Joye, and BenoîT Libert. A new framework for privacy-preserving aggregation of time-series data. *ACM Trans. Inf. Syst. Secur.*, 18(3), mar 2016.

[BLV+21] Lukas Burkhalter, Hidde Lycklama, Alexander Viand, Nicolas Küchler, and Anwar Hithnawi. Rofl: Attestable robustness for secure federated learning. *CoRR*, 2021.

[BSK+19] Keith Bonawitz, Fariborz Salehi, Jakub Konečný, Brendan McMahan, and Marco Gruteser. Federated learning with autotuned communication-efficient secure aggregation. In *2019 53rd Asilomar Conference on Signals, Systems, and Computers*, 2019.

[BSW11] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In Yuval Ishai, editor, *Theory of Cryptography*. Springer Berlin Heidelberg, 2011.

[BT20] Constance Beguier and Eric W. Tramel. Safer: Sparse secure aggregation for federated learning, 2020.

[BTL+21] Carlo Brunetta, Georgia Tsaloli, Bei Liang, Gustavo Banegas, and Aikaterini Mitrokotsa. Non-interactive, secure verifiable aggregation for decentralized, privacy-preserving learning. In *Information Security and Privacy*, Cham, 2021. Springer International Publishing.

[BVH+20a] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How to backdoor federated learning. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, 2020.

[BVH+20b] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How to backdoor federated learning. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*. PMLR, 2020.

[CCMT09] Claude Castelluccia, Aldar C-F. Chan, Einar Mykletun, and Gene Tsudik. Efficient and provably secure aggregation of encrypted data in wireless sensor networks. *ACM Trans. Sen. Netw.*, 5, 2009.

[CDE+18] Ronald Cramer, Ivan Damgård, Daniel Escudero, Peter Scholl, and Chaoping Xing. Spd $\mathbb{Z}_{2^k}$: Efficient mpc mod 2^k for dishonest majority. In *Advances in Cryptology – CRYPTO 2018*. Springer International Publishing, 2018.

[CGB17] Henry Corrigan-Gibbs and Dan Boneh. Prio: Private, robust, and scalable computation of aggregate statistics. In *Proceedings of the 14th USENIX Conference on Networked Systems Design and Implementation*, NSDI'17. USENIX Association, 2017.

[CMS13] Ricardo J. G. B. Campello, Davoud Moulavi, and Joerg Sander. Density-based clustering based on hierarchical density estimates. In *Advances in Knowledge Discovery and Data Mining*. Springer Berlin Heidelberg, 2013.

[CMT05] C. Castelluccia, E. Mykletun, and G. Tsudik. Efficient aggregation of encrypted data in wireless sensor networks. In *The Second Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services*, 2005.

[CO18] Michele Ciampi and Claudio Orlandi. Combining private set-intersection with secure two-party computation. In *Security and Cryptography for Networks*. Springer International Publishing, 2018.

[Cou18] Geoffroy Couteau. New protocols for secure equality test and comparison. In *Applied Cryptography and Network Security*. Springer International Publishing, 2018.

[CPTPH21] Sylvain Chatel, Apostolos Pyrgelis, Juan Ramón Troncoso-Pastoriza, and Jean-Pierre Hubaux. Sok: Privacy-preserving collaborative tree-based model learning. *Proceedings on Privacy Enhancing Technologies*, 2021, 2021.

[CSS12] T-H. Hubert Chan, Elaine Shi, and Dawn Song. Privacy-preserving stream aggregation with fault tolerance. In *Financial Cryptography and Data Security*, pages 200–214. Springer Berlin Heidelberg, 2012.

[CYD20] Zhou Chuanxin, Sun Yi, and Wang Degang. Federated learning with gaussian differential privacy. In *Proceedings of the 2020 2nd International Conference on Robotics, Intelligent Control and Artificial Intelligence*, RICAI 2020, New York, NY, USA, 2020. Association for Computing Machinery.

[DA16] Tassos Dimitriou and Mohamad Khattar Awad. Secure and scalable aggregation in the smart grid resilient against malicious entities. *Ad Hoc Networks*, 50, 2016.

[DCSW20] Ye Dong, Xiaojun Chen, Liyan Shen, and Dakui Wang. Eastfly: Efficient and secure ternary federated learning. *Computers & Security*, 94:101824, 2020.

[Dep21] Webank AI Department. Fate. https://fate.fedai.org/, 2021.

[DGBL+16] Nathan Dowlin, Ran Gilad-Bachrach, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16. JMLR.org, 2016.

[DH06] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Trans. Inf. Theor.*, 2006.

[DHCP21] Phan The Duy, Huynh Nhat Hao, Huynh Minh Chu, and Van-Hau Pham. A secure and privacy preserving federated learning approach for iot intrusion detection system. In *Network and System Security*, Cham, 2021. Springer International Publishing.

[DN03] Irit Dinur and Kobbi Nissim. Revealing information while preserving privacy. In *Proceedings of the Twenty-Second ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '03. Association for Computing Machinery, 2003.

[DOS18] Ivan Damgård, Claudio Orlandi, and Mark Simkin. Yet another compiler for active security or: Efficient mpc over arbitrary rings. In *Advances in Cryptology – CRYPTO 2018*, volume 10992 of *Lecture Notes in Computer Science*. Springer, 2018.

[DOT18] Pratish Datta, Tatsuaki Okamoto, and Junichi Tomida. Full-hiding (unbounded) multi-input inner product functional encryption from the k-linear assumption. In *Public-Key Cryptography – PKC 2018*. Springer International Publishing, 2018.

[DP21] Anirban Das and Stacy Patterson. Multi-tier federated learning for vertically partitioned data. In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021.

[Dwo06] Cynthia Dwork. Differential privacy. In *Automa, Languages and Programming*. Springer Berlin Heidelberg, 2006.

[EA20] Ahmed Roushdy Elkordy and Amir Salman Avestimehr. Secure aggregation with heterogeneous quantization in federated learning. *CoRR*, abs/2009.14388, 2020.

[ET12] Zekeriya Erkin and Gene Tsudik. Private computation of spatial and temporal power consumption with smart meters. In *Applied Cryptography and Network Security*, pages 561–577. Springer Berlin Heidelberg, 2012.

[FJR15] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, New York, NY, USA, 2015. Association for Computing Machinery.

[FMLF21] Joaquín Delgado Fernández, Sergio Potenciano Menci, Charles Lee, and Gilbert Fridgen. Secure federated learning for residential short term load forecasting. *CoRR*, abs/2111.09248, 2021.

[FMM+21] Hossein Fereidooni, Samuel Marchal, Markus Miettinen, Azalia Mirhoseini, Helen Möllering, Thien Duc Nguyen, Phillip Rieger, Ahmad-Reza Sadeghi, Thomas Schneider, Hossein Yalame, and Shaza Zeitouni. Safelearn: Secure aggregation for private federated learning. In *2021 IEEE Security and Privacy Workshops (SPW)*, 2021.

[Fre12] David Mandell Freeman. Improved security for linearly homomorphic signatures: A generic framework. In *Public Key Cryptography – PKC 2012*. Springer Berlin Heidelberg, 2012.

[FYB18] Clement Fung, Chris J. M. Yoon, and Ivan Beschastnikh. Mitigating sybils in federated learning poisoning. *CoRR*, abs/1808.04866, 2018.

[GGG+14] Shafi Goldwasser, S. Dov Gordon, Vipul Goyal, Abhishek Jain, Jonathan Katz, Feng-Hao Liu, Amit Sahai, Elaine Shi, and Hong-Sheng Zhou. Multi-input functional encryption. In *Advances in Cryptology – EUROCRYPT 2014*. Springer Berlin Heidelberg, 2014.

[GHG+21] Jiqiang Gao, Boyu Hou, Xiaojie Guo, Zheli Liu, Ying Zhang, Kai Chen, and Jin Li. Secure aggregation is insecure: Category inference attack on federated learning. *IEEE Transactions on Dependable and Secure Computing*, 2021.

[GLL+21] Xiaojie Guo, Zheli Liu, Jin Li, Jiqiang Gao, Boyu Hou, Changyu Dong, and Thar Baker. Verifl: Communication-efficient and fast verifiable aggregation for federated learning. *IEEE Transactions on Information Forensics and Security*, 16, 2021.

[GWY+18] Karan Ganju, Qi Wang, Wei Yang, Carl A. Gunter, and Nikita Borisov. Property inference attacks on fully connected neural networks using

permutation invariant representations. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, CCS '18, New York, NY, USA, 2018. Association for Computing Machinery.

[HE03] Lingxuan Hu and D. Evans. Secure aggregation for wireless networks. In *2003 Symposium on Applications and the Internet Workshops, 2003. Proceedings.*, 2003.

[HKKH21] Changhee Hahn, Hodong Kim, Minjae Kim, and Junbeom Hur. Versa: Verifiable secure aggregation for cross-device federated learning. *IEEE Transactions on Dependable and Secure Computing*, pages 1–1, 2021.

[HKR+18] Andrew Hard, Chloé M Kiddon, Daniel Ramage, Francoise Beaufays, Hubert Eichner, Kanishka Rao, Rajiv Mathews, and Sean Augenstein. Federated learning for mobile keyboard prediction, 2018.

[HMSY21] Aditya Hegde, Helen Möllering, Thomas Schneider, and Hossein Yalame. Sok: Efficient privacy-preserving clustering. Cryptology ePrint Archive, Report 2021/809, 2021.

[HTGW18] Ehsan Hesamifard, Hassan Takabi, Mehdi Ghasemi, and Rebecca N. Wright. Privacy-preserving machine learning as a service. *Proceedings on Privacy Enhancing Technologies*, 2018, 2018.

[iA93] Shun ichi Amari. Backpropagation and stochastic gradient descent method. *Neurocomputing*, 5, 1993.

[IKOS06] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Cryptography from anonymity. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 239–248, 2006.

[JGP+21] Zoe L. Jiang, Hui Guo, Yijian Pan, Yang Liu, Xuan Wang, and Jun Zhang. Secure neural network in federated learning with model aggregation under multiple keys. In *2021 8th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)/2021 7th IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom)*, 2021.

[JL13] Marc Joye and Benoît Libert. A scalable scheme for privacy-preserving aggregation of time-series data. In Ahmad-Reza Sadeghi, editor, *Financial Cryptography and Data Security*. Springer Berlin Heidelberg, 2013.

[JNMALC22] Tayyebeh Jahani-Nezhad, Mohammad Ali Maddah-Ali, Songze Li, and Giuseppe Caire. Swiftagg: Communication-efficient and dropout-resistant secure aggregation for federated learning with worst-case security guarantees, 2022.

[KDK11] Klaus Kursawe, George Danezis, and Markulf Kohlweiss. Privacy-friendly aggregation for the smart-grid. In *Privacy Enhancing Technologies*. Springer Berlin Heidelberg, 2011.

[KLS21] Peter Kairouz, Ziyu Liu, and Thomas Steinke. The distributed discrete gaussian mechanism for federated learning with secure aggregation. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*. PMLR, 2021.

[KMA+19] Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista A. Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, Rafael G. L. D'Oliveira, Salim El Rouayheb, David Evans, Josh Gardner, Zachary Garrett, Adrià Gascón, Badih Ghazi, Phillip B. Gibbons, Marco Gruteser, Zaïd Harchaoui, Chaoyang He, Lie He, Zhouyuan Huo, Ben Hutchinson, Justin Hsu, Martin Jaggi, Tara Javidi, Gauri Joshi, Mikhail Khodak, Jakub Konečný, Aleksandra Korolova, Farinaz Koushanfar, Sanmi Koyejo, Tancrède Lepoint, Yang Liu, Prateek Mittal, Mehryar Mohri, Richard Nock, Ayfer Özgür, Rasmus Pagh, Mariana Raykova, Hang Qi, Daniel Ramage, Ramesh Raskar, Dawn Song, Weikang Song, Sebastian U. Stich, Ziteng Sun, Ananda Theertha Suresh, Florian Tramèr, Praneeth Vepakomma, Jianyu Wang, Li Xiong, Zheng Xu, Qiang Yang, Felix X. Yu, Han Yu, and Sen Zhao. Advances and open problems in federated learning. *CoRR*, abs/1912.04977, 2019.

[KOB21] Ferhat Karakoç, Melek Önen, and Zeki Bilgin. Secure aggregation against malicious users. In *Proceedings of the 26th ACM Symposium on Access Control Models and Technologies*, SACMAT '21. Association for Computing Machinery, 2021.

[KRKR20] Swanand Kadhe, Nived Rajaraman, Onur Ozan Koyluoglu, and Kannan Ramchandran. Fastsecagg: Scalable secure aggregation for privacy-preserving federated learning. *CoRR*, abs/2009.11248, 2020.

[KShS12] Benjamin Kreuter, Abhi Shelat, and Chih hao Shen. Billion-gate secure computation with malicious adversaries. In *21st USENIX Security Symposium (USENIX Security 12)*, 2012.

[KSRW04] T. Kohno, A. Stubblefield, A.D. Rubin, and D.S. Wallach. Analysis of an electronic voting system. In *IEEE Symposium on Security and Privacy, 2004. Proceedings. 2004*, 2004.

[KTC20] Youssef Khazbak, Tianxiang Tan, and Guohong Cao. Mlguard: Mitigating poisoning attacks in privacy preserving distributed collaborative learning. In *2020 29th International Conference on Computer Communications and Networks (ICCCN)*, 2020.

[LCV19] Changchang Liu, Supriyo Chakraborty, and Dinesh Verma. *Secure Model Fusion for Distributed Learning Using Partial Homomorphic Encryption*. Springer International Publishing, 2019.

[LEM14a] Iraklis Leontiadis, Kaoutar Elkhiyaoui, and Refik Molva. Private and dynamic time-series data aggregation with trust relaxation. In *Cryptology and Network Security*, pages 305–320. Springer International Publishing, 2014.

[LEM14b] Iraklis Leontiadis, Kaoutar Elkhiyaoui, and Refik Molva. Private and dynamic time-series data aggregation with trust relaxation. In *Cryptology and Network Security*. Springer International Publishing, 2014.

[LHCH20] Zhaorui Li, Zhicong Huang, Chaochao Chen, and Cheng Hong. Quantification of the leakage in federated learning, 2020.

[LWH19] Qinbin Li, Zeyi Wen, and Bingsheng He. Federated learning systems: Vision, hype and reality for data privacy and protection. *CoRR*, abs/1907.09693, 2019.

[MASN21] Ahmed Moustafa, Muhammad Asad, Saima Shaukat, and Alexander Norta. Ppcsa: Partial participation-based compressed and secure aggregation in federated learning. In *Advanced Information Networking and Applications*. Springer International Publishing, 2021.

[MHK+21] Fan Mo, Hamed Haddadi, Kleomenis Katevas, Eduard Marin, Diego Perino, and Nicolas Kourtellis. Ppfl: Privacy-preserving federated learning with trusted execution environments. In *Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '21, New York, NY, USA, 2021. Association for Computing Machinery.

[MMR+17] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*. PMLR, 2017.

[MPP+21] Viraaji Mothukuri, Reza M. Parizi, Seyedamin Pouriyeh, Yan Huang, Ali Dehghantanha, and Gautam Srivastava. A survey on security and privacy of federated learning. *Future Generation Computer Systems*, 115, 2021.

[MSDCS19] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. Exploiting unintended feature leakage in collaborative learning. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 691–706, 2019.

[NMZ+21] John Nguyen, Kshitiz Malik, Hongyuan Zhan, Ashkan Yousefpour, Michael Rabbat, Mani Malek Esmaeili, and Dzmitry Huba. Federated learning with buffered asynchronous aggregation. *CoRR*, abs/2106.06639, 2021.

[NP05] Moni Naor and Benny Pinkas. Computationally secure oblivious transfer. *Journal of Cryptology*, 18, 2005.

[NRY+21] Thien Duc Nguyen, Phillip Rieger, Hossein Yalame, Helen Möllering, Hossein Fereidooni, Samuel Marchal, Markus Miettinen, Azalia Mirhoseini, Ahmad-Reza Sadeghi, Thomas Schneider, and Shaza Zeitouni. FLGUARD: secure and private federated learning. *CoRR*, abs/2101.02281, 2021.

[NSH19] Milad Nasr, Reza Shokri, and Amir Houmansadr. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In *2019 IEEE Symposium on Security and Privacy (SP)*, 2019.

[ÖM07] Melek Önen and Refik Molva. Secure data aggregation with multiple encryption. In *Wireless Sensor Networks*, pages 117–132. Springer Berlin Heidelberg, 2007.

[Ope21] OpenMined. Pysyft. https://github.com/OpenMined/PySyft, 2021.

[Pad21] PaddlePaddle. Paddlefl. https://github.com/PaddlePaddle/PaddleFL, 2021.

[PAH+18] Le Trieu Phong, Yoshinori Aono, Takuya Hayashi, Lihua Wang, and Shiho Moriai. Privacy-preserving deep learning via additively homomorphic encryption. *IEEE Transactions on Information Forensics and Security*, 13, 2018.

[Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology — EUROCRYPT '99*, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

[Ped92] Torben Pryds Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Advances in Cryptology — CRYPTO '91*. Springer Berlin Heidelberg, 1992.

[PFA21] Dario Pasquini, Danilo Francati, and Giuseppe Ateniese. Eluding secure aggregation in federated learning via model inconsistency. *CoRR*, abs/2111.07380, 2021.

[Pol78] J. Pollard. Monte carlo methods for index computation (). *Mathematics of Computation*, 32:918–924, 1978.

[Riv97] Ronald L. Rivest. All-or-nothing encryption and the package transform. In *Fast Software Encryption*. Springer Berlin Heidelberg, 1997.

[RL89] R.M. Roth and A. Lempel. On mds codes via cauchy matrices. *IEEE Transactions on Information Theory*, 35(6), 1989.

[Rub96] Frank Rubin. One-time pad cryptography. *Cryptologia*, 1996.

[SAG+21] Jinhyun So, Ramy E. Ali, Basak Guler, Jiantao Jiao, and Salman Avestimehr. Securing secure aggregation: Mitigating multi-round privacy leakage in federated learning. *CoRR*, abs/2106.03328, 2021.

[SAGA21] Jinhyun So, Ramy E. Ali, Basak Güler, and Amir Salman Avestimehr. Secure aggregation for buffered asynchronous federated learning. *CoRR*, abs/2110.02177, 2021.

[SCR+11] Elaine Shi, T.-H Chan, Eleanor Rieffel, Richard Chow, and Dawn Song. Privacy-preserving aggregation of time-series data. volume 2, 01 2011.

[SGA21a] Jinhyun So, Başak Göler, and A. Salman Avestimehr. Byzantine-resilient secure federated learning. *IEEE Journal on Selected Areas in Communications*, 39, 2021.

[SGA21b] Jinhyun So, Başak Güler, and A. Salman Avestimehr. Turbo-aggregate: Breaking the quadratic aggregation barrier in secure federated learning. *IEEE Journal on Selected Areas in Information Theory*, 2, 2021.

[Sha79] Adi Shamir. How to share a secret. *Commun. ACM*, 1979.

[SMH21] Thomas Sandholm, Sayandev Mukherjee, and Bernardo A. Huberman. SAFE: secure aggregation with failover and encryption. *CoRR*, abs/2108.05475, 2021.

[SS11] Abhi Shelat and Chih-Hao Shen. Two-output secure computation with malicious adversaries. *IACR Cryptol. ePrint Arch.*, 2011:533, 2011.

[SSSS17a] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*, 2017.

[SSSS17b] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*, 2017.

[SSV+21] Timothy Stevens, Christian Skalka, Christelle Vincent, John Ring, Samuel Clark, and Joseph Near. Efficient differentially private secure aggregation for federated learning via hardness of learning with errors. *CoRR*, abs/2112.06872, 2021.

[STL20] Mohamed Seif, Ravi Tandon, and Ming Li. Wireless federated learning with local differential privacy. In *2020 IEEE International Symposium on Information Theory (ISIT)*, 2020.

[STS16] Shiqi Shen, Shruti Tople, and Prateek Saxena. Auror: Defending against poisoning attacks in collaborative deep learning systems. In *Proceedings of the 32nd Annual Conference on Computer Security Applications*, ACSAC '16. Association for Computing Machinery, 2016.

[SWRH20] Lushan Song, Haoqi Wu, Wenqiang Ruan, and Weili Han. Sok: Training machine learning models over multiple sources with privacy preservation. *CoRR*, abs/2012.03386, 2020.

[TBA+19] Stacey Truex, Nathalie Baracaldo, Ali Anwar, Thomas Steinke, Heiko Ludwig, Rui Zhang, and Yi Zhou. A hybrid approach to privacy-preserving federated learning. In *Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security*, AISec'19. Association for Computing Machinery, 2019.

[TF19] Aleksei Triastcyn and Boi Faltings. Federated learning with bayesian differential privacy. In *2019 IEEE International Conference on Big Data (Big Data)*, 2019.

[TLB+21] Georgia Tsaloli, Bei Liang, Carlo Brunetta, Gustavo Banegas, and Aikaterini Mitrokotsa. DEVA: Decentralized, verifiable secure aggregation for privacy-preserving learning. In *Information Security*, Cham, 2021. Springer International Publishing.

[TNW+21] Minxue Tang, Xuefei Ning, Yitu Wang, Yu Wang, and Yiran Chen. Fedgp: Correlation-based active client selection for heterogeneous federated learning. abs/2103.13822, 2021.

[VNP+20] Anamaria Vizitiu, Cosmin Ioan Nită, Andrei Puiu, Constantin Suciu, and Lucian Mihai Itu. Applying deep neural networks over homomorphic encrypted medical data. *Computational and Mathematical Methods in Medicine*, 2020, 2020.

[VXK21] Raj Kiriti Velicheti, Derek Xia, and Oluwasanmi Koyejo. Secure byzantine-robust distributed learning via clustering. *CoRR*, abs/2110.02940, 2021.

[WGA06] D. Westhoff, J. Girao, and M. Acharya. Concealed data aggregation for reverse multicast traffic in sensor networks: Encryption, key distribution, and routing adaptation. *IEEE Transactions on Mobile Computing*, 5, 2006.

[WGC19] Sameer Wagh, Divya Gupta, and Nishanth Chandran. Securenn: 3-party secure computation for neural network training. *Proceedings on Privacy Enhancing Technologies*, 2019, 2019.

[WLW+09] Rui Wang, Yong Fuga Li, XiaoFeng Wang, Haixu Tang, and Xiaoyong Zhou. Learning your identity and disease from research papers: Information leaks in genome wide association study. In *Proceedings of the 16th ACM Conference on Computer and Communications Security*, CCS '09. Association for Computing Machinery, 2009.

[WPX+20] Danye Wu, Miao Pan, Zhiwei Xu, Yujun Zhang, and Zhu Han. Towards efficient secure aggregation for model update in federated learning. In *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*, 2020.

[WTB+20] Sameer Wagh, Shruti Tople, Fabrice Benhamouda, Eyal Kushilevitz, Prateek Mittal, and Tal Rabin. FALCON: honest-majority maliciously secure framework for private deep learning. *CoRR*, abs/2004.02229, 2020.

[XBZ+19] Runhua Xu, Nathalie Baracaldo, Yi Zhou, Ali Anwar, and Heiko Ludwig. Hybridalpha: An efficient approach for privacy-preserving federated learning. In *Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security*, AISec'19. Association for Computing Machinery, 2019.

[XHCL20] Chulin Xie, Keli Huang, Pin-Yu Chen, and Bo Li. Dba: Distributed backdoor attacks against federated learning. In *International Conference on Learning Representations*, 2020.

[XLL+20] Guowen Xu, Hongwei Li, Sen Liu, Kan Yang, and Xiaodong Lin. Verifynet: Secure and verifiable federated learning. *IEEE Transactions on Information Forensics and Security*, 15, 2020.

[YAE+18] Timothy Yang, Galen Andrew, Hubert Eichner, Haicheng Sun, Wei Li, Nicholas Kong, Daniel Ramage, and Françoise Beaufays. Applied federated learning: Improving google keyboard query suggestions. *CoRR*, abs/1812.02903, 2018.

[YLCT19] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *ACM Trans. Intell. Syst. Technol.*, 10, 2019.

[YSH+21] Chien-Sheng Yang, Jinhyun So, Chaoyang He, Songze Li, Qian Yu, and Salman Avestimehr. Lightsecagg: Rethinking secure aggregation in federated learning. *CoRR*, abs/2109.14236, 2021.

[YWW21] Ge Yang, Shaowei Wang, and Haijie Wang. Federated learning with personalized local differential privacy. In *2021 IEEE 6th International Conference on Computer and Communication Systems (ICCCS)*, 2021.

[ZFW+20] Xianglong Zhang, Anmin Fu, Huaqun Wang, Chunyi Zhou, and Zhenzhu Chen. A privacy-preserving and verifiable federated learning scheme. In *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, 2020.

[ZJF+21] Lingchen Zhao, Jianlin Jiang, Bo Feng, Qian Wang, Chao Shen, and Qi Li. Sear: Secure and efficient aggregation for byzantine-robust federated learning. *IEEE Transactions on Dependable and Secure Computing*, pages 1–1, 2021.

[ZLH19] Ligeng Zhu, Zhijian Liu, and Song Han. Deep leakage from gradients. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[ZLX+20] Chengliang Zhang, Suyi Li, Junzhe Xia, Wei Wang, Feng Yan, and Yang Liu. Batchcrypt: Efficient homomorphic encryption for cross-silo federated learning. In *2020 USENIX Annual Technical Conference (USENIX ATC 20)*. USENIX Association, 2020.

[ZLYM21] Zhuosheng Zhang, Jiarui Li, Shucheng Yu, and Christian Makaya. Safelearning: Enable backdoor detectability in federated learning with secure aggregation. *CoRR*, abs/2102.02402, 2021.

[ZWC+21] Yuhui Zhang, Zhiwei Wang, Jiangfeng Cao, Rui Hou, and Dan Meng. Shufflefl: Gradient-preserving federated learning using trusted execution environment. In *Proceedings of the 18th ACM International Conference on Computing Frontiers*, CF '21, New York, NY, USA, 2021. Association for Computing Machinery.

A AHE SCHEMES FOR SECURE AGGREGATION

A.1 Shi-Chan-Rieffel-Chow-Song Scheme

SCRCS scheme [SCR+11] is the first Key AHE scheme used for secure aggregation. It guarantees *Aggregator Obliviousness* based on Decisional Diffie Hellman (DDH) assumption. The three algorithms ($AHE.Setup$, $AHE.Enc$, and $AHE.Agg$) are defined as follows:

- $(pp, \{k_i\}_{i\in[1,..,n]}, k_0) \leftarrow AHE.Setup(\lambda)$: Given a security parameter λ, it chooses a generator $g \in \mathbb{G}$ where $\mathbb{G}$ is a cyclic group of prime order p for which Decisional Diffie-Hellman is hard. Additionally, it defines the hash function $H : \mathbb{Z} \rightarrow \mathbb{G}$. It also generates n random secrets $k_1, ..., k_n \in \mathbb{Z}_n$ and $k_0 = -\sum_1^N s_i$. It outputs the public parameters $pp = (g, H)$, the secrets keys of each user $\{k_i\}_{i\in[1,..,n]}$, and the secret key of the aggregator k_0.

- $c_{i,\tau} \leftarrow AHE.Enc(pp, k_i, \tau, m_{i,\tau})$:
$$c_{i,\tau} \leftarrow g^{m_{i,\tau}} \cdot H(\tau)^{k_i}$$
- $\sum_1^n m_{i,\tau} \leftarrow AHE.Agg(pp, k_0, \tau, \{c_{i,\tau}\}_{i\in[1,..,n]})$:
$$c_\tau \leftarrow \prod_1^n c_{i,\tau} = g^{\sum_1^N m_{i,\tau}} \cdot H(\tau)^{\sum_1^N k_i}$$
$$V \leftarrow H(\tau)^{k_0} \cdot c_\tau = g^{\sum_1^n m_{i,\tau}} \mod n$$
Then it compute the discrete logarithm base g of V to obtain $\sum_1^n m_{i,\tau} \bmod p$. For efficient computation of the discrete logarithm using Pollard's method [Pol78], the output $\sum_1^n m_{i,\tau}$ should be a small number.

For the prove of correctness and security of this scheme, refer to [SCR+11].

A.2 Joye-Libert Scheme

JL scheme [JL13] is another AHE scheme for SA which was designed as an improvement of the SCRCS scheme. JL scheme has a simpler decryption function as it does not require the computation of the discrete logarithm in a group in which the DDH assumption hold. The JL scheme guarantees *Aggregator Obliviousness* based on Decision Composite Residuosity (DCR) assumption [Pai99]. It defines the three algorithms (*AHE.Setup*, *AHE.Enc*, and *AHE.Agg*) as follows:

- $(pp, \{k_i\}_{i\in[1,..,n]}, k_0) \leftarrow AHE.Setup(\lambda)$: Given a security parameter λ, it generates randomly two equal-size prime numbers p and q and sets $N = pq$. Then, it defines a cryptographic hash function $H : \mathbb{Z} \rightarrow \mathbb{Z}^*_{N^2}$. It randomly generates n secret keys $\{k_i\}_{1..n} \in \mathbb{Z}_{N^2}$ and sets $k_0 = -\sum_1^n k_i$. It outputs the public parameters $pp = (N, H)$, the secrets keys of each user $\{k_i\}_{i\in[1,..,n]}$, and the secret key of the aggregator k_0.
- $c_{i,\tau} \leftarrow AHE.Enc(pp, k_i, \tau, m_{i,\tau})$:
$$c_{i,\tau} \leftarrow (1 + m_{i,\tau}N) \cdot H(\tau)^{k_i} \mod N^2$$
- $\sum_1^n m_{i,\tau} \leftarrow AHE.Agg(pp, k_0, \tau, \{c_{i,\tau}\}_{i\in[1,..,n]})$:
$$c_\tau \leftarrow \prod_1^n c_{i,\tau} = (1 + N\sum_1^n m_{i,\tau}) \cdot H(\tau)^{\sum_1^n k_i} \mod N^2$$
$$\frac{H(\tau)^{k_0} c_\tau - 1}{N} = \sum_1^n m_{i,\tau} \mod N$$

For the prove of correctness and security of this scheme, refer to [JL13].

B LAYERED MASKING VARIANT

Another variant of masking-based secure aggregation is the layered masking [CCMT09, CMT05, ÖM07, WGA06]. In this type of masking scheme, the users are assumed to have network connectivity with each others. So, the users arrange in a tree structure. In the **SA**.**Setup** phase, the ones that are at a distance h hops from each other, shares the same keys (h being a security parameter). Each user representing a node in the tree runs a secure aggregation process with its children. In **SA**.**Protect**, each child node masks its input with the sum of the keys it holds using *Masking.Mask*. It then sends it to its parent node. In **SA**.**Agg**, the parent sums the masked inputs received from all its children and then removes the layers of the masks that corresponds to their keys using *Masking.UnMask*. The same process is repeated from bottom to up for each parent node until the aggregated value reaches the root of the tree at which the final layers are removed. Castelluccia et al. [CMT05] applied a specific version of this scheme when $h = \infty$. Önen et al. [ÖM07] later generalized this scheme.

C FAULT-TOLERANT SA WITH MASKING

Masking is one of the lightest techniques in terms of communication and computational overhead. It also offers, a descent security against collusion attacks. However, using Masking for secure aggregation is inefficient when the data owners are mobile or when the network is highly disrupted. This is because the failure of one user providing its input results in the failure of the entire secure aggregation operation. To bypass this limitation, Shamir Secret Sharing is integrated with Masking which derive a new fault tolerance secure aggregation technique [BIK+17, BBG+20, SGA21b]. This technique uses the DC-net variant of masking. In the **SA**.**Setup** phase, the users agree on mutual seeds using Deffie-Hellman similar to the case in standard masking-based SA. However, for FT-Masking, users additionally using the t-out-of-n Shamir's Secret Sharing [Sha79] to share their Deffie-Helmen secret key. Using this approach, masks of dropped users can be recovered as long as t users are still alive. The outcome owner can reconstruct the DH secret key of the missing users and consequently compute their masks. While this solves the problem of dropped users, it causes a new security problem. If a user delays in sending its masked input to the aggregator, the aggregator may consider it as dropped and thus asks for reconstructing its masks. Later when the masked input is received, the outcome owner is able to unmask its input. To solve this problem, a double masking technique is used in the **SA**.**Protect**. In detail, each user adds another layer of masking using a randomly generated mask $b_{i,\tau}$:
$$c_{i,\tau} \leftarrow m_{i,\tau} + k_{i,\tau} + b_{i,\tau}$$
where $k_{i,\tau}$ is computed as in the equation 1. This new mask is generated from a random generated seed which is also shared using Shamir's Secret Sharing with all other users. In **SA**.**Agg**, the aggregator first collects t shares of the seed of each mask $b_{i,\tau}$ for every alive user $\mathcal{U}_i$ and reconstruct it. Then it get t shares of the Deffie-Helmen's secret key of the dropped users and thus reconstruct the missing masks $k_{j,\tau}$ for every dropped data owner $\mathcal{U}_j$. Consider $\mathcal{X}$ and $\mathcal{Y}$ the set of remaining and dropped users respectively (all the operation are mod $R = nR_u$ where $[0, R_u]$ is the range for the input values of each user).

$$\begin{aligned}
&\sum_{\mathcal{U}_i\in\mathcal{X}} c_{i,\tau} - \sum_{\mathcal{U}_i\in\mathcal{X}} b_{i,\tau} + \sum_{\mathcal{U}_j\in\mathcal{Y}} k_{i,\tau} \\
&= \sum_{\mathcal{U}_i\in\mathcal{X}} m_{i,\tau} + \sum_{\mathcal{U}_i\in\mathcal{X}} k_{i,\tau} + \sum_{\mathcal{U}_i\in\mathcal{X}} b_{i,\tau} - \sum_{\mathcal{U}_i\in\mathcal{X}} b_{i,\tau} + \sum_{\mathcal{U}_j\in\mathcal{Y}} k_{i,\tau} \\
&= \sum_{\mathcal{U}_i\in\mathcal{X}} m_{i,\tau} + \cancelto{0}{\sum_{\mathcal{U}_i\in\mathcal{X}\cup\mathcal{Y}} k_{i,\tau}} \\
&= \sum_1^n m_{i,\tau}
\end{aligned}$$