IACR Transactions on Cryptographic Hardware and Embedded Systems
ISSN 2569-2925, Vol. 2026, No. 1, pp. 185–224. DOI:10.46586/tches.v2026.i1.185-224

WW-FL: Secure and Private Large-Scale Federated Learning

Felix Marx[1], Thomas Schneider[1], Ajith Suresh[2], Tobias Wehrle[1], Christian Weinert[3] and Hossein Yalame[1]

[1] Technical University of Darmstadt, Darmstadt, Germany
{lastname}@encrypto.cs.tu-darmstadt.de
[2] Technology Innovation Institute, Abu Dhabi, UAE
{firstname.lastname}@tii.ae
[3] Royal Holloway, University of London, London, UK
{firstname.lastname}@rhul.ac.uk

Abstract. Federated learning (FL) is an efficient approach for large-scale distributed machine learning that promises data privacy by keeping training data on client devices. However, recent research has uncovered vulnerabilities in FL, impacting both security and privacy through poisoning attacks and the potential disclosure of sensitive information in individual model updates as well as the aggregated global model. This paper explores the inadequacies of existing FL protection measures when applied independently, and the challenges of creating effective compositions.
Addressing these issues, we propose WW-FL, an innovative framework that combines secure multi-party computation (MPC) with hierarchical FL to guarantee data and global model privacy. One notable feature of WW-FL is its capability to prevent malicious clients from directly poisoning model parameters, confining them to less destructive data poisoning attacks. We furthermore provide a PyTorch-based FL implementation integrated with Meta's CrypTen MPC framework to systematically measure the performance and robustness of WW-FL. Our extensive evaluation demonstrates that WW-FL is a promising solution for secure and private large-scale federated learning.

Keywords: Federated Learning · Hierarchical FL · MPC · Poisoning

1 Introduction

Federated learning (FL) [MMR+17] is a revolutionary approach for large-scale distributed machine learning that emerged in the past decade and has had a profound impact on both academia [YZH22] and industry [RHL+20, LBT+20, HLS+20]. A World Wide Web Consortium (W3C) community group currently works towards establishing FL-related Web standards [W3C25]. FL shares some objectives with privacy-preserving machine learning (PPML) [GDL+16, MZ17, NC23, SBBE25], but differs in its approach to training a model: In PPML, training on the *entire set of available data* is either outsourced to a remote party or done collaboratively while ensuring data privacy. In contrast, FL involves multiple rounds of training, where each round consists of selected clients *locally training a model on their private data* and a central party performing aggregation of these models to update a global model.

Initially, FL appeared to indeed offer privacy since the training data remains on the client devices, and only model updates (so-called gradients) are transmitted. However, subsequent research has demonstrated that these gradients still reveal a significant amount of information, enabling the central aggregator to infer sensitive details about the clients'

Licensed under Creative Commons License CC-BY 4.0.
Received: 2025-07-15 Accepted: 2025-09-15 Published: 2026-01-16

private training data [GBDM20, ZSE+24, WBS+24, DHW+25]. To address this issue, *secure aggregation* (SA) schemes [BIK+17, BBG+20, MOJC23] were proposed, where only the aggregated result is revealed to the central party, effectively concealing the individual gradients. Despite this, FL still poses several complex challenges that demand careful consideration [ZBA+25].

The first challenge relates to serious privacy vulnerabilities that have been identified when using secure aggregation with a *single* aggregator [BDS+22, FGC+22, WGF+22, SAG+23, BDS+23b, ZSE+24]: Recent studies have demonstrated the effectiveness of privacy attacks even when utilizing secure aggregation techniques across many clients, showing that such schemes are insufficient [PFA22, BDS+23a]. These vulnerabilities largely stem from the imbalance of power held by the central aggregator, who can introduce inconsistencies in the global model, manipulate client selection, and exert disproportionate influence over the training process.

These growing concerns over privacy with a single central aggregator hinder the use of standard FL, and even FL with SA, for global training tasks as sharing sensitive data across jurisdictions is often problematic legally. For example, the Court of Justice of the European Union (CJEU) with its famous "Schrems II" judgment has ruled the "Privacy Shield" agreement between the EU and the US for exchanging personal data invalid [Tra20]. Until agreeing on the most recent "Data Privacy Framework" [U.S22], numerous attempts to set up data sharing have failed, including the "Safe Harbour" principles [WA16]. Recent works therefore suggest a distributed aggregator setup using secure multi-party computation (MPC) to securely distribute aggregation tasks among multiple servers, ensuring privacy despite potential collusion among a subset of them [FMM+21, RSWP23, GMS+23, BMP+24, TXLZ25].

The second (and closely related) challenge is the current lack of *global model privacy*. Initially, FL focused on enhancing user participation and improving model accuracy rather than safeguarding the model itself. However, with the widespread adoption of FL in various industries, such as the medical sector [RHL+20], the need for privacy protection becomes apparent. Unfortunately, unrestricted access to the aggregated model allows extraction of traces of original training data, necessitating *global model privacy* [PFA22, BDS+23a, ZSE+24, WCH+24]. So far, only a small number of works have focused on preserving the privacy of the global model in FL [MG19, SPT+21]. Most of these works use homomorphic encryption (HE) schemes [AAUC18], which primarily address semi-honest corruptions. Moreover, several works do not consider the collusion of clients with the aggregator and thereby provide a weaker notion of security. Furthermore, organizations coordinating FL may have a vested interest in keeping the global model private—even from the contributing clients—as revealing the model could pose business risks [DXD+23].

The third challenge arises when corrupted clients intentionally manipulate the locally trained model to reduce the system's accuracy or introduce backdoors to extract sensitive information in future rounds [WSR+20, XHCL20, ZPS+22]. This method of *model poisoning* is more potent than *data poisoning* [SHKR22, XFG25], which involves manipulating the dataset. Although numerous defenses have been proposed against model poisoning [SH21, CFLG21, ZCJG22], the most effective remain impractical for large-scale deployment due to their high computation and communication costs [NRC+22, RSWP23, TXLZ25]. Additionally, recent research has introduced even more potent model poisoning strategies that can circumvent state-of-the-art defenses [SH21, GGP24, XFG25]. For instance, PoisonedFL [XFG25] breaks eight state-of-the-art defenses without any knowledge of the honest clients' models, highlighting the urgent need for new defense mechanisms. To address corrupted clients, we limit their attack capabilities to data poisoning rather than the more severe model poisoning. This allows us to implement more moderate defenses compared to existing schemes (cf. §5 for details).

Besides these three challenges, a large-scale, worldwide FL deployment needs to account for client heterogeneity in terms of bandwidth, computational power, jurisdiction, and trust assumptions. As we will elaborate in §1.1, trust in government and private entities varies widely, influencing how data is shared and used.

Although various works have addressed the aforementioned issues individually or in combination, a comprehensive solution remains elusive. Therefore, we propose a unified framework called WW-FL, which enables secure and private distributed machine learning at scale. Departing from traditional FL, we introduce the concept of a *trust zone*, allowing data to be shared securely within a trusted legal jurisdiction and leveraging privacy-enhancing technologies like MPC. This approach supports edge nodes with limited resources and encourages wider participation, enhancing the system's overall privacy and performance.

1.1 A Regulatory and Trust Perspective

Beyond the critical security, privacy, and technical deployment challenges that exist in FL and have been discussed so far, it is equally important to consider the regulatory landscape and the varying degrees of trust among participating entities. Specifically, WW-FL aims to address the *heterogeneity in trust* across entities distributed geographically [OEC22,Sta22]. This variation is not only regional but also institutional, with differing levels of trust in, for example, law enforcement versus judicial bodies [OEC22].

When it comes to private companies, some countries have implemented strict laws to ensure the ethical behavior of entities operating within their borders. Non-compliance can result in substantial fines and even imprisonment [Hus23, Hil23]. Recent examples include Meta receiving a 1.2 billion Euro fine from regulators in Ireland [Hus23] and Didi Global facing a 1.19 billion Euro penalty from the Cyberspace Administration of China [Hil23]. Several companies including well-known TalkTalk and Target either lost their customers' data in major data breaches or were accused of not keeping customer information safe. This led to a lack of trust among their clients and a significant drop in revenue [Eri22,Hib22].

Given this disparity in trust, consider a consortium comprising three companies interested in training a machine learning (ML) model for medical diagnostics. Numerous studies have demonstrated the effectiveness of ML models in diagnosing various diseases, such as lung cancer, tumor detection, and liver segmentation [RHJ+24]. However, these studies emphasize the need for diverse patient data. Simultaneously, the sensitivity and privacy of the training data must be preserved.

To address this, the consortium may collaborate with various federal and local government agencies to obtain relevant training data and potentially compensating data providers. However, since the model is trained on sensitive medical data, the consortium members do not want the trained model to be disclosed to anyone. Additionally, the consortium should have the ability to use the trained model for inference while preserving privacy. This approach not only enhances the system's privacy by safeguarding against breaches that might affect individual companies but also safeguards the financial investments of these companies by preventing unauthorized use of their trained model by third parties.

At first glance, implementing a PPML solution may appear simplistic. In PPML, data owners securely distribute their data among the three companies within the consortium, ensuring that no individual company can access the data in the clear. The consortium employs MPC techniques to privately train a model on the shared data. However, this approach has several drawbacks. One significant drawback is the substantial computational power required within the consortium to handle the training process, particularly when dealing with millions of data owners in a global deployment. Furthermore, this approach necessitates a high level of trust from all data owners, as they must share their data among the consortium's member companies. This trust requirement can be a significant hurdle to adoption. Additionally, there may be situations where data owners are willing to share their data but cannot do so due to government regulations preventing data from leaving their jurisdiction. This restriction not only limits data availability but also renders this approach unsuitable for our specific use case.

The next approach we can consider is using a traditional FL with secure aggregation. While this method has the potential to address the scalability problem, it comes with a significant drawback. In this approach, the model during training is distributed to the data owners. Although there are works like [FTP+21, SPT+21, XHX+23], which protect the privacy of the global model using homomorphic encryption techniques, they may not be well-suited for our cross-device setting with a large number of data owners. Additionally, these techniques necessitate interaction among the clients and require extra measures for handling dropouts.

In light of the limitations of current methods, we make an adjustment to the concept of FL. Specifically, we expand the notion of a *trust zone*, allowing data to be moved from the data owner to this larger, still-trusted area. For instance, a user in our scenario can securely share their data among three entities: the police, the court, and a private company, all operating within the same legal jurisdiction as the data owner. These selected entities will collaboratively train the model using privacy-enhancing technologies like MPC and HE. Furthermore, this approach has the added benefit of supporting edge nodes with limited computational resources, allowing them to contribute to the system. Importantly, since data owners can be assured that their data remains within their designated trust zone, this will likely encourage more users to participate, leading to a larger pool of training data and, ultimately, a better-trained model while guaranteeing global privacy.

Finally, it is important to note that the concept of a trust-zone can differ depending on the specific region or context. For example, in one country, it may be relatively straightforward to identify three entities that data owners trust not to collude with each other. In contrast, there may be situations where it is challenging to find three or four entities that can provide an absolute guarantee of non-collusion. However, in such cases, data owners may be willing to accept a weaker assurance that not all entities will collude. We can accommodate this heterogeneity in trust by performing PPML training independently for each of these scenarios, using the appropriate MPC protocol, and then combining their results through secure aggregation using global servers.

1.2 Our Contributions

In this paper, we initiate a discussion on novel approaches towards secure and private large-scale (cross-device) federated learning to address the challenges outlined above. Towards this goal, we propose a unified framework called "worldwide-FL" (WW-FL). Our framework is based on a novel abstraction that also captures existing hierarchical FL architectures in a *hybrid* manner as visualized in Fig. 1 and detailed in §3.5.

Briefly, in WW-FL, clients use secret-sharing to securely outsource training data to distributed *cluster servers* based on MPC. The clients then can leave and return sporadically to provide more training data – this makes our framework robust against real-world issues like drop-outs, requires no interaction between clients, and resource-constrained (mobile or edge) devices only have very little workload. The models trained by the cluster servers with MPC-based PPML techniques are then aggregated across training clusters using one or multiple levels of distributed aggregators. For secure distributed aggregation, we again utilize MPC. After aggregation, models are not converted to plaintext but are returned to the training clusters for the next training iteration in secret-shared form. When training is completed, known secure inference protocols can be used to allow private queries [NC23, MWCB23] in a controlled (potentially rate-limited) way. Our architecture addresses issues with single aggregators by relying on a *distributed aggregator* and notably achieves *global model privacy* due to a novel combination of FL with PPML techniques.

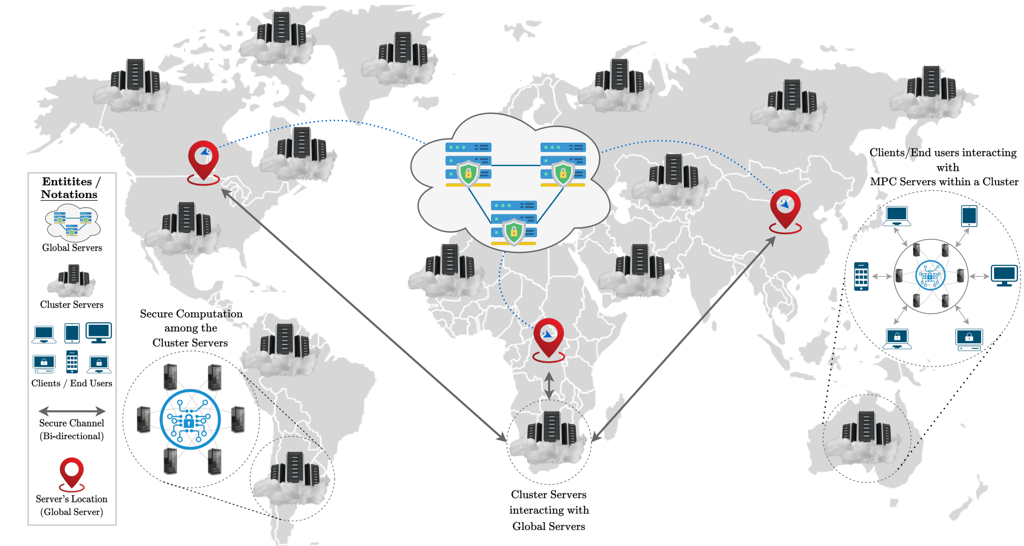


Figure 1: Visual representation of deploying our secure world-wide FL (WW-FL) framework. In this illustration, clients connect to local MPC clusters, which in turn are connected to three global servers (this can be varied as per requirements) strategically positioned across continents. Fig. 2 provides a detailed illustration of WW-FL, showcasing the roles of involved entities.

We observe that a unique property of our framework is the strictly limited attack surface: similar to PPML, malicious clients are restricted to weaker data-poisoning attacks as there is no possibility to access and manipulate the model itself. As defending against data poisoning in PPML is highly non-trivial, we show that state-of-the-art data-poisoning attacks in the suggested hierarchical configuration are less effective than in plain FL. Furthermore, we consider different robust aggregation schemes to further mitigate the effect of such attacks. For this, we additionally propose new heuristics that improve the efficiency in MPC. Specifically, we propose a lightweight variant of the "Trimmed Mean" [YCRB18] defense for our setting with only data-poisoning attacks.

Finally, we systematically evaluate possible instantiations of WW-FL in terms of performance (both in terms of achievable accuracy as well as overhead due to MPC) and resilience against poisoning attacks. For this, we implement a prototype based on PyTorch and Meta's CrypTen framework [KVH+21] for instantiating MPC. We then evaluate the performance when training neural networks for standard image classification tasks in realistic network settings and using GPU-accelerated AWS EC2 instances. This way, we show that the computation and communication overhead induced by MPC is feasible even for large-scale deployments. For example, one round of training LeNet privately with security against semi-honest servers on the MNIST dataset for 5 epochs with batch size 80 takes less than 5 minutes for a cluster with 10 clients, and global aggregation among 10 clusters takes less than 1 second. Moreover, we find that the impact of MPC-induced inaccuracies is less than 0.1%, and models trained with WW-FL converge significantly faster than in plain FL.

In Tab. 1, we summarize how WW-FL distinguishes itself from related works. With respect to existing works on collaborative learning while also achieving global model privacy, we experimentally demonstrate superior scalability and performance. For example, compared to the state-of-the-art framework Hercules [XHX+23], which is based on multi-party homomorphic encryption, we demonstrate convolutional neural network training with 20× more clients, 4× less communication, and 250× less storage overhead.

In summary, we provide the following contributions:

- We initiate a discussion to explore why novel approaches are necessary to jointly address concerns regarding security, privacy, and scalability in FL.

- We propose a new generic (hierarchical) FL framework called WW-FL that achieves global model privacy, supports resource-constrained mobile or edge devices, and significantly limits the attack surface for malicious clients to data poisoning.

- We systematically evaluate the performance of WW-FL instantiations as well as its resilience against state-of-the-art poisoning attacks. Additionally, we provide a prototype implementation and evaluation of WW-FL on standard image classification tasks.

- We experimentally compare WW-FL to state-of-the-art collaborative learning frameworks with global model privacy, demonstrating superior scalability and performance.

2 Preliminaries and Related Work

This section provides a brief background on privacy-enhancing technologies like MPC and HE, and an introduction to FL, along with a concise overview of related works. We provide a comparative positioning in Tab. 1 showing that no prior work simultaneously provides global model privacy and protection against malicious clients.

Secure Multi-Party Computation (MPC) MPC [Yao86, GMW87] enables a set of mutually distrusting parties to evaluate a public function $f()$ on their private data while preserving input data privacy. The corruption among the parties is often modelled via an *adversary* that may try to take control of the corrupted parties and coordinate their actions. There are various orthogonal aspects of adversarial corruption like honest vs. dishonest majority, semi-honest vs. malicious corruption, etc. [EKR18, Lin20]. MPC is practically efficient when run among a small number of computing parties [PS20, KVH+21, KS22, KPRS22, HSW+25]. In our prototype implementation[1] of WW-FL, we utilize Meta's CrypTen framework [KVH+21], which for efficiency reasons operates in a semi-honest two-party setting with a trusted third "helper" party that generates correlated randomness [RWT+18, CCPS19, PSSY21].

Homomorphic Encryption (HE) HE schemes [RAD78, Gen09] enable computation on encrypted data without the need for decryption. Additively homomorphic encryption (AHE) is a widely used method that allows for the generation of a new ciphertext representing the sum of multiple plaintexts through operations on their corresponding ciphertexts [Pai99]. In scenarios involving multiple parties, recent multi-party homomorphic encryption (MHE) schemes can reduce the communication complexity of MPC, but have high computation overhead [MTPBH21]. Furthermore, they are often secure only against semi-honest corruptions, whereas [CMS+23] is an exception. A comprehensive survey of various HE schemes is given in [AAUC18].

Differential Privacy (DP) The concept of DP [DR14] is based on the idea of adding noise to data in order to reduce information leakage when sharing it, while still allowing for meaningful computations to be carried out on the data. Though DP techniques offer some protection against attacks in federated learning, they also reduce the utility of the data and the resulting ML model. Additionally, achieving robust and accurate models necessitates significant privacy budgets, leaving the actual level of privacy achieved in practice uncertain [OA22]. We refer to [DP20] for more details on DP.

Federated Learning (FL) Unlike conventional PPML techniques, FL [KMY+16, MMR+17] enables the training of an ML model on distributed data by allowing each device to train the model locally using its own data. In each iteration, these local models are sent to a central server (also called an aggregator), where they are *aggregated* to form a global model. At a high level, an FL scheme iterates through the following steps:

[1] Our framework is adaptable to various settings and threat models, as discussed in §3.3. We built our implementation on CrypTen due to its maturity and ML-friendly design.

- The global server $\mathcal{S}$ sends the current global model W_t to a selected subset of n out of N clients.

- Each selected client C^i, $i \in [n]$ utilizes its own local training data D^i for E epochs to fine-tune the global model and obtains an updated local model w^i_{t+1}:

$$w^i_{t+1} \leftarrow W_t - \eta_{C^i} \frac{\partial L(W_t, B_{i,e})}{\partial W_t},$$

where L is a loss function, η_{C^i} is the clients' learning rate, and $B_{i,e} \subseteq D^i$ is a batch drawn from D^i in epoch e, where $e \in [E]$. The local model updates w^i_{t+1} are then sent back to $\mathcal{S}$.

- $\mathcal{S}$ employs an aggregation rule f_{agg} to combine the received local model updates w^i_{t+1}, resulting in a global model W_{t+1}, which will serve as the starting point for the next iteration:

$$W_{t+1} \leftarrow W_t - \eta_{\mathsf{S}} \cdot f_{agg}(w^1_{t+1}, \ldots, w^n_{t+1}),$$

where η_{S} is the server's learning rate.

The above procedure is repeated until a predefined stopping criterion is satisfied, e.g., a specified number of training iterations or a specific level of accuracy.

Secure Aggregation (SA**)** Standard FL's privacy issues [GBDM20, ZSE+24, WBS+24, DHW+25] led to single-server secure aggregation protocols (Single $\mathcal{S}$) [BIK+17, BBG+20, MOJC23] that use masking techniques or HE [Rub96, PAH+18, MG19]. However, complete privacy for the global model remains elusive, posing risks if clients and aggregators collude. Moreover, recent research has shown that a single malicious server can reconstruct individual training data points from users' local models even when using secure aggregation protocols [PFA22, BDS+23a, ZSE+24, WCH+24]. Such vulnerabilities against colluding parties are addressed by multi-server SA (Multi $\mathcal{S}$) [FMM+21, DCL+21, NRC+22, RSWP23, TXLZ25] and techniques like multi-party homomorphic encryption (MHE) [FTP+21, SPT+21, XHX+23], albeit with scalability challenges. Works like [DCL+21] and [NRC+22] combine HE and MPC techniques to achieve private and robust FL, albeit at the cost of considerable computation and communication overhead in cross-device scenarios comprising hundreds to thousands of clients [SHKR22]. In WW-FL, we employ a multi-server aggregation scheme, and mitigate scalability issues of existing works, especially for a heterogeneous cross-device setting.

Collaborative Learning with Global Model Privacy Several prior works focus explicitly on preserving the privacy of the global model during collaborative learning. These approaches primarily rely on HE, particularly multi-party homomorphic encryption (MHE) [MTPBH21], to securely aggregate model updates while maintaining privacy across multiple clients. MHE enables distributed and fully decentralized training through direct interactions among data owners in settings where non-colluding servers are unavailable. Spindle [FTP+21] employs MHE to support secure distributed training, specifically for logistic regression models. However, Spindle is limited to linear models, which restricts its applicability to more complex architectures. Poseidon [SPT+21] extends MHE to 3-layer neural networks by leveraging multi-party CKKS [MTPBH21]. In this framework, multiple entities collaboratively generate a shared secret key, enabling encrypted computations under a common public key. To maintain computational efficiency, Poseidon replaces non-polynomial functions, such as the commonly used ReLU activation function, with polynomial approximations. However, these polynomial approximations introduce accuracy trade-offs that may significantly impact model performance. To address these computational inefficiencies, Hercules [XHX+23] was proposed as an enhancement of Poseidon. Hercules introduces optimized parallel homomorphic operations, including SIMD-enabled

Table 1: Comparison of WW-FL and previous works. Notations: $\mathcal{S}$ – Aggregation Server(s), $\mathcal{C}$ – Client, GM – Global Model, LM – Local Model, AHE/MHE – Additively/Multi-Party Homomorphic Encryption, ZKP – Zero-Knowledge Proof, MPC – Secure Multi-party Computation, DP – Differential Privacy. In terms of privacy, we focus on the protection of the global model on both server and client side as well as the protection of individual client models against a curious or malicious server. Given the extensive literature, this comparison focuses on one representative work per category.

Categories	Representative Work(s)	Privacy Method	Privacy ($\mathcal{S}$)		Privacy ($\mathcal{C}$)	Defense	Cross Device	No Client Interaction	Dropout Handling
			GM	LM	GM				
Aggregation (Plain)	[MMR+17]	–	✗	✗	✗	✗	✓	✓	✓
Aggregation (Robust)	[BMGS17]	–	✗	✗	✗	✓	✗	✓	✓
	[CFLG21, ZWP+23]	–	✗	✗	✗	✓	✓	✓	✓
Secure Aggregation (SINGLE $\mathcal{S}$)	[BIK+17]	Masking	✗	✓	✗	✗	✗	✗	✓
	[MG19]	AHE	✗	✓	✓	✗	✗	✓	✓
	[GJS+23]	Masking+ZKP	✗	✓	✗	✓	✗	✗	✓
	[MYP24, LBG+25]	Masking	✗	✓	✗	✓	✗	✗	✓
Secure Aggregation (MULTI $\mathcal{S}$)	[FMM+21]	MPC	✓	✓	✗	✗	✓	✓	✓
	[FTP+21]*	MHE	✓	✓	✓	✗	✗	✗	✗
	[SPT+21, XHX+23]*	MHE	✓	✓	✓	✗	✗	✗	✗
	[NRC+22]	MPC	✓	✓	✗	✓	✗	✓	✓
	[RSWP23]	MPC	✓	✓	✗	✗‡	✓	✓	✓
	[TXLZ25]	MPC	✓	✓	✗	✗‡⋆	✓	✓	✓
Hierarchical FL (HFL)	[BEG+19]	Masking	✗	✓	✗	✗	✓	✗	✓
	[WXL+21]	–	✗	✗	✗	✗	✓	✓	✓
	[Yan21]	DP	✗	✓	✗	✗	✓	✓	✓
	[FHC+24]	–	✗	✗	✗	✗	✓	✓	✓
	[WHZ+25]	Masking	✗	✓	✗	✓	✓	✓	✓
Worldwide FL (WW-FL)	**This work**	MPC†	✓	✓	✓	✓	✓	✓	✓

∗ Represents collaborative learning in the N-party setting, rather than the Federated Learning (FL) setting. SPINDLE [FTP+21] supports only generalized linear models and lacks protocols for training complex models like NNs; POSEIDON [SPT+21] and HERCULES [XHX+23] use activation function approximations, reducing final accuracy.
‡ Limited to defenses that operate independently on each client's gradient; excludes aggregation methods like trimmed mean or median [YCRB18].
⋆ Cannot detect malformed gradients that lie within the norm bound (e.g., shrunk gradients).
† WW-FL is a generic design that can be implemented with various secure computation protocols, such as MPC or HE.

matrix computations and improved polynomial approximations for RELU. While Hercules offers higher computational performance than Poseidon, it continues to suffer from accuracy degradation due to the use of polynomial approximations for non-linear layers.

Despite these advancements, all three MHE-based frameworks – Spindle [FTP+21], Poseidon [SPT+21], and Hercules [XHX+23] – exhibit notable limitations. The reliance on polynomial approximations for non-linear functions, particularly ReLU, inherently reduces model accuracy [RRK+20]. Moreover, these frameworks lack dedicated robustness mechanisms against poisoning attacks, without which a single malicious participant can substantially degrade the global model accuracy [FCJG20]. Scalability also remains a critical limitation: the experiments in these works typically involve no more than 50 parties, raising concerns about their feasibility in larger, real-world collaborative scenarios. Additionally, these approaches require interactive key setup among clients for encrypting each model and cooperative decryption of the final aggregated model, which introduces further complexity and communication overhead.

Another significant bottleneck in HE-based frameworks is their high storage complexity. For example, training on the MNIST dataset using a lightweight 3-layer fully connected model, Poseidon suffers from an exceptionally high local storage burden – up to 314GB per client. This results from its packing encryption method, which replicates encrypted values and inserts zero-padding to match layer sizes, depending on the number of neurons in each hidden layer. Such replication leads to multiple ciphertext copies and excessive padding, rendering the approach inefficient for client-side deployment and impractical for large-scale systems. While Hercules improves computational efficiency, it still incurs a 4.9GB local storage overhead per client – potentially prohibitive for resource-constrained devices. In contrast, WW-FL avoids this overhead entirely by eliminating ciphertext storage on the client side: clients keep their training data in plaintext, while servers hold secret shares of both training data and model parameters. The size of secret shares might not always exactly match that of the plaintext; they are typically larger by a small constant factor (e.g., 2×-3×), depending on the secret-sharing scheme and number of servers. This design makes our framework not only faster, but also significantly lighter in terms of storage requirements, particularly benefiting resource-constrained clients such as IoT devices.

Hierarchical Federated Learning (HFL) Another approach for FL, called hierarchical FL (HFL) addresses scalability and heterogeneity issues in real-world systems [BEG+19, WWCJ22]. In HFL, aggregation occurs at multiple levels, forming a hierarchical structure in which the aggregated values from one level serve as inputs for the above level. This procedure eventually reaches the top level, where the final model is aggregated [DLR+21, WXL+21]. Most of the works in the HFL setting have primarily focused on improving scalability and communication [FHC+24, WHZ+25]. An exception is the work by [Yan21], which proposed methods to ensure the privacy of individual client updates in HFL, allowing for secure aggregation. However, their approach does not address global model privacy or provide robustness against malicious users, which we provide in WW-FL.

Poisoning & Robust Aggregation One potential drawback of FL over PPML techniques is that FL increases the attack surface for malicious actors: in FL, each user trains their own model locally, allowing them to manipulate their model [BCMC19, SH21]. To counter these *poisoning* attacks, various robust aggregation methods have been proposed, in which local models are analyzed and appropriate measures such as outlier removal or norm scaling are applied [YCRB18, LXC+19, CFLG21, BMP+24]. See [SHKR22, GGP24, ZBA+25] for a comprehensive overview of different robust aggregations.

Model Extraction Attacks In model extraction attacks, an adversary leverages black-box access to a model's outputs to deduce its structure without prior knowledge [LPLW24, ZBA+25]. While existing privacy-preserving FL frameworks using MPC or HE protect

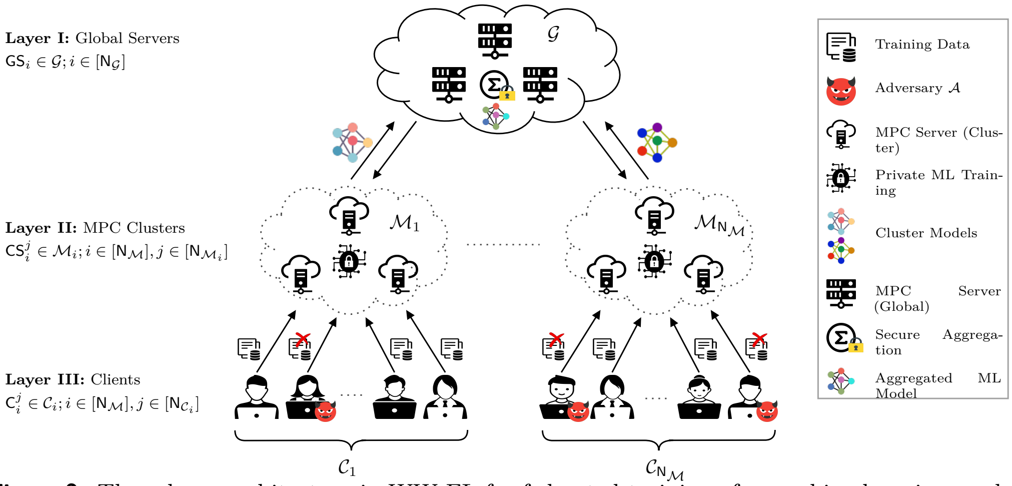


Figure 2: Three-layer architecture in WW-FL for federated training of a machine learning model. The number of servers and clients shown is illustrative and not fixed. See §3.6 for further details.

data privacy, they do not prevent such attacks, as the adversary still has access to model outputs. We consider model extraction prevention orthogonal to private inference in FL and suggest incorporating methods like rate limiting and differential privacy [ZLD+25] for future work.

3 The WW-FL Framework

We now present the details of our WW-FL framework, which addresses multiple key requirements in FL: global model privacy, scalability, support for resource-constrained (mobile/edge) devices, reduction of attack surface, ability to defend against data poisoning, and high levels of user engagement. Our framework captures various proposed architectures for FL in a unified abstraction. We illustrate our framework in Fig. 2 and detail the underlying training algorithm in Alg. 1. The notations used in WW-FL are listed in Tab. 2.

Table 2: Notations used in our WW-FL framework.

Notation	Description
$\mathcal{G}$	Set of Global MPC Servers.
$\mathcal{M}_i$	Set of MPC Cluster Servers in the i-th cluster.
$\mathcal{C}_i^{\mathsf{U}}$	Set of Clients in the i-th cluster.
$\mathcal{C}_i$	Selected Clients in the i-th cluster.
N_s	Size of the set $s \in \{\mathcal{G}, \mathcal{M}_i, \mathcal{C}_j\}$.
$\mathsf{N}_{\mathcal{M}}$	Total number of clusters.
W_t	Global model available at round t.
GS_i	Layer I MPC Global Server; $\mathsf{GS}_i \in \mathcal{G}$.
CS_i^j	Layer II MPC Cluster Server; $\mathsf{CS}_i^j \in \mathcal{M}_i$. Here, $i \in [\mathsf{N}_{\mathcal{M}}], j \in [\mathsf{N}_{\mathcal{M}_i}]$
C_i^j	Layer III Client; $\mathsf{C}_i^j \in \mathcal{C}_i$, $i \in [\mathsf{N}_{\mathcal{M}}], j \in [\mathsf{N}_{\mathcal{C}_i}]$
$\langle\cdot\rangle_s$	Secret sharing semantics for $s \in \{\mathcal{G}, \mathcal{M}_j\}$.

3.1 WW-FL Architecture

Our WW-FL framework is based on a three-layer[2] architecture and extends the established hierarchical FL (HFL) paradigm [BEG+19, WXL+21, Yan21]. In HFL, clients are initially

[2] Additional MPC Cluster layers can be easily incorporated depending on the scale of deployment.

organized into clusters, and their local models are aggregated at the cluster level; these cluster-level models are then further aggregated globally, resulting in an additional layer of aggregation. We adopt the hierarchical approach in WW-FL for two main reasons: 1) to efficiently realize our distinct model training approach for global model privacy, and 2) to facilitate large-scale deployments across heterogeneous clients. However, there is one key difference: at the cluster level, WW-FL does not perform aggregation of models but private training on collective client data. This differs from both traditional and hierarchical FL since the data is not kept solely on the client devices. Nevertheless, WW-FL ensures that the data remains within the trusted domain of the client, allowing for accurate modeling of trust relationships between clients in real-world situations, such as trust among members of the same region or country [MG06]. Furthermore, this type of hierarchy is ubiquitous in real-world scenarios such as P2P gaming, organizations, and network infrastructures [SARK02, HV19, LSPJ20].

We provide details for each layer in WW-FL next. We focus on the sequence of operations performed by the entities in our architecture (cf. Fig. 2) for a training over T iterations while considering necessary MPC protocols and setup requirements. To make our design generic, we use the $\mathcal{F}_{\mathsf{MPC}}$-hybrid model, a standard way to abstract MPC protocols. Concrete instantiations are given in Tab. 3 in §3.2.

Layer III: Clients This layer is composed of $\mathsf{N}_{\mathcal{M}}$ distinct sets of clients $\mathcal{C}_i^{\mathsf{U}}$ (with $i \in [\mathsf{N}_{\mathcal{M}}]$), called *clusters*, which are formed based on specific criteria of the application (e.g., European Union (EU) member states for the EU smart metering scheme [CK13, Com14]). Like standard FL, only a random subset of clients $\mathcal{C}_i \subseteq \mathcal{C}_i^{\mathsf{U}}$ will be selected by the training algorithm in an iteration $t \in [1, T]$. This subset selection is performed by the MPC clusters in Layer II using the Sample algorithm, ensuring that the selection process adheres to the desired security guarantees.

Table 3: MPC functionalities used in WW-FL.

Algorithm	Input(s)	Description
Share	D, V	Generates secret shares of data D as per target V's sharing semantics ($\langle D\rangle_V$) with $V \in \{\mathcal{C}, \mathcal{M}, \mathcal{G}\}$.
Reshare	$\langle D\rangle_U, V$	Converts secret shares of data $\langle D\rangle$ from source U's sharing semantics to target V.
Agg	$\{\langle W^j\rangle\}$	Performs secure aggregation over j secret-shared ML models.
Train	$\langle W\rangle, \langle D\rangle$	Performs PPML Training on ML model W using the secret-shared data $\langle D\rangle$.
Predict	$\langle W\rangle, \langle Q\rangle$	Performs PPML Inference on secret-shared ML model $\langle W\rangle$ using the secret-shared query $\langle Q\rangle$.
Reveal	$\langle D\rangle_U, V$	Reconstructs secret-shared data $\langle D\rangle$ towards members in target set V.
Sample	$\mathcal{C}^{\mathsf{U}}$	Selects a subset of clients $\mathcal{C}$ from $\mathcal{C}^{\mathsf{U}}$.
TM-List	$\mathcal{W}, \alpha$	Performs Trimmed Mean and returns 2α outlier values (top and bottom α) for each index position in elements of $\mathcal{W}$.
TopK-Hitter	$\mathcal{U}, \Gamma$	Returns list of Γ values that occur most frequently in $\mathcal{U}$.

During iteration t, client $\mathsf{C}_i^j \in \mathcal{C}_i$ (with $j \in [\mathsf{N}_{\mathcal{C}_i}]$) holding data $D_t^{\mathsf{C}_i^j}$ uses the Share protocol (cf. Tab. 3) to securely distribute its data to a set of MPC cluster servers $\mathcal{M}_i$. $\mathcal{M}_i$ are a representative group of high-performance servers that are trusted by their clients not to collude entirely, though some level of collusion among a subset of servers may occur. WW-FL allows clients to share their input and then leave at any time. They can also rejoin the system later and provide additional data in the next iteration they get selected. Hence, the clusters $\mathcal{C}_i^{\mathsf{U}}$ are dynamic and change with each iteration.

Our method differs from the standard concept of "data residing at the clients" in FL, but we expect it to not negatively impact user engagement as data remains within the users' trust zone. Additionally, the reduced computational load allows resource-constrained devices to train complex models and eliminates the need for a shared-key setup among clients, simplifying dropout handling.

Layer II: MPC Clusters The second layer consists of $\mathsf{N}_{\mathcal{M}}$ sets of distributed training servers $\mathcal{M}_i$ (with $i \in \mathsf{N}_{\mathcal{M}}$), called *MPC clusters*, with each $\mathcal{M}_i$ corresponding to cluster $\mathcal{C}_i^{\mathsf{U}}$ in Layer III. In iteration t, Layer I servers (denoted by $\mathcal{G}$) initiate ML training by sharing the current global model W_{t-1} among the servers in $\mathcal{M}_i$. As will be discussed in §3.1, W_{t-1} is also in a secret-shared form among $\mathcal{G}$, represented by $\langle W_{t-1} \rangle_{\mathcal{G}}$. To account for varying availability and trustworthiness of servers across regions, MPC clusters in WW-FL may use different MPC configurations and differ, e.g., in their corruption threshold and security model [EKR18]. Therefore, $\mathcal{G}$ uses the Reshare protocol (cf. Tab. 3) to convert the secret shares of $\langle W_{t-1} \rangle_{\mathcal{G}}$ to those of $\mathcal{M}_i$, i.e., $\langle W_{t-1} \rangle_{\mathcal{M}_i}$.

Given $\langle W_{t-1} \rangle_{\mathcal{M}_i}$, the servers in $\mathcal{M}_i$ use the Train protocol (cf. Tab. 3) to run MPC-based PPML for private ML training [KVH+21, KS22] on the cumulative secret-shared data from all clients in the cluster $\mathcal{C}_i$, denoted by $\langle D_t \rangle_{\mathcal{M}_i}$. This data may include data from the same cluster in the previous iteration. Furthermore, by utilizing a larger pool of training data, we can leverage the known benefits of batching, resulting in faster convergence [GDG+17, BCN18]. After completing training, the servers in $\mathcal{M}_i$ run Reshare to secret-share the updated model with the Layer I servers, i.e., $\langle W_t^i \rangle_{\mathcal{G}}$.

To preserve the system's integrity, the servers for each MPC cluster must be chosen with care to ensure clients are willing to share their data among the servers and that not all servers are colluding. One option is to build non-profit partnerships, such as in the MOC alliance [ZIC+21], where organizations with mutual distrust can securely co-locate servers in the same data center with high-speed network connections. Alternatively, trusted entities like government organizations with limited infrastructure can host their servers in confidential cloud computing environments [RCF+21]. If a client lacks trust in the WW-FL servers in its current cluster, the client can propose a new cluster with a partner that provides a non-colluding server.

Layer I: Global Servers The top layer consists of a set of MPC servers $\mathcal{G}$, named *Global Servers*, that *securely* aggregate trained models from all the MPC clusters in Layer II, similarly to a standard FL scheme with a distributed aggregator [FMM+21, RSWP23, GMS+23, BMP+24, TXLZ25]. Given the locally trained models W_t^i for $i \in [\mathsf{N}_{\mathcal{M}}]$, the servers in $\mathcal{G}$ execute the secure aggregation protocol Agg [MOJC23] to compute the updated global model in secret-shared form, i.e., $\langle W_t \rangle_{\mathcal{G}}$. The global servers in $\mathcal{G}$ use the Reshare protocol to distribute the aggregated model W_t to each of the Layer II clusters $\mathcal{M}_i$ for the next iteration $t+1$.

The distributed aggregator model, as outlined in §1, circumvents the possibility of several privacy attacks that were recently presented for single-server aggregators even when relying on maliciously secure aggregation protocols [BDS+22, FGC+22, WGF+22, SAG+23, BDS+23b, ZSE+24]; for example, by excluding certain inputs from the computation and thus influencing the *correctness* (but not the security) of the protocol, a malicious single server can mount *privacy* attacks that allow to infer information about particular participants and their training data. In contrast, in our distributed aggregator setting, even when having only semi-honest protocol security, such attacks are detectable as long as at least one of the servers behaves honestly.

WW-FL – The Complete Picture Alg. 1 (FL-Train) provides the training algorithm in WW-FL. We detail the necessary MPC functionalities along with the WW-FL training functionality $\mathcal{F}_{\mathsf{WW\text{-}FL}}$ (Alg. 3) in §3.3. Although WW-FL has a 3-layer architecture, it can easily accommodate more levels of hierarchy depending on the size of the deployment. For this, additional layers of MPC clusters can be added between layers I and II, with the clusters performing secure aggregation instead of PPML training.

Algorithm 1 FL-Train: Training in WW-FL

Actors: $\mathcal{G}, \mathcal{M}, \mathcal{C}$ # $\mathcal{M} = \bigcup_i \mathcal{M}_i$, $\mathcal{C} = \bigcup_i \mathcal{C}_i$; $i \in \mathsf{N}_{\mathcal{M}}$

Input: W_0, $\{D^{\mathsf{C}}\}_{\mathsf{C} \in \mathcal{C}}$ # W_0 – initial model, D^{C} – client C's data

Output: $\langle W_T \rangle$ # W_T – global model after T iterations

initialize: $\langle W_0 \rangle_{\mathcal{G}} \leftarrow \textsc{Share}(W_0, \mathcal{G})$

for each training iteration $t \in [1, T]$ **do**

  for all $i \in \mathsf{N}_{\mathcal{M}}$ **do** # in parallel

    $\langle W_{t-1} \rangle_{\mathcal{M}_i} \leftarrow \mathcal{G}.\textsc{Reshare}(\langle W_{t-1} \rangle_{\mathcal{G}}, \mathcal{M}_i)$ # global servers $\mathcal{G}$ reshare W_{t-1} with cluster servers

    $\mathcal{C}_i \leftarrow \mathcal{M}_i.\textsc{Sample}(\mathcal{C}_i^{\mathsf{U}}, t)$ # $\mathcal{C}_i^{\mathsf{U}}$ – total clients in i-th cluster

    for all $j \in [\mathsf{N}_{\mathcal{C}_i}]$ **do** # in parallel

      $\langle D_t^{\mathsf{C}_i^j} \rangle_{\mathcal{M}_i} \leftarrow \mathsf{C}_i^j.\textsc{Share}(D_t^{\mathsf{C}_i^j}, \mathcal{M}_i)$ # $D_t^{\mathsf{C}_i^j}$ – client C_i^j's data in iteration t

    end for

    $\langle D_t \rangle_{\mathcal{M}_i} \leftarrow \bigcup_{j \in [\mathsf{N}_{\mathcal{C}_i}]} \langle D_t^{\mathsf{C}_i^j} \rangle_{\mathcal{M}_i} \bigcup \langle D_{t-1} \rangle_{\mathcal{M}_i}$ # $D_0 = \emptyset$

    $\langle W_t^i \rangle_{\mathcal{M}_i} \leftarrow \mathcal{M}_i.\textsc{Train}(\langle W_{t-1} \rangle_{\mathcal{M}_i}, \langle D_t \rangle_{\mathcal{M}_i})$

    $\langle W_t^i \rangle_{\mathcal{G}} \leftarrow \mathcal{M}_i.\textsc{Reshare}(\langle W_t^i \rangle_{\mathcal{M}_i}, \mathcal{G})$

  end for

  $\langle W_t \rangle_{\mathcal{G}} \leftarrow \mathcal{G}.\textsc{Agg}(\{\langle W_t^i \rangle_{\mathcal{G}}\}_{i \in [\mathsf{N}_{\mathcal{M}}]})$

end for

Existing schemes for global model privacy, such as [RSWP23] and [TXLZ25], only protect the model from either clients or aggregator servers, leaving the possibility of collusion especially in a cross-device setting. WW-FL addresses this issue by keeping the global model in secret-shared form, ensuring that no single entity or group of colluding entities (up to an allowed corruption threshold) can access the model. This provides a stronger notion of privacy and also protects against unauthorized use or misuse, such as a client disclosing the trained model to another organization for further training or commercial use. For scenarios demanding disclosure of the global model to authorized entities, such as consortium members, the global servers can utilize the Reveal protocol.

In §3.6, we provide a detailed explanation of the FL-Train algorithm for a scenario involving three Layer II MPC clusters, each using a different protocol: two-party [DSZ15], three-party [AFL+16,MR18], and four-party [KPRS22]. The global servers are instantiated using a two-party protocol with a helper, based on CrypTen [KVH+21].

3.2 MPC Functionalities in WW-FL

The MPC functionalities utilized in WW-FL are listed in Tab. 3 and discussed in the following. While Share and Reshare are used for generating the secret shares as per the underlying MPC semantics, Reveal is used to reconstruct the secret towards a designated party. The Train and Predict functionalities correspond to PPML training and inference protocols, respectively. Similarly, Agg denotes the secure aggregation functionality in FL (cf. §2). In WW-FL, these functionalities are realized using Meta's CrypTen framework, in which two semi-honest MPC servers carry out the computation with the help of a trusted third party that provides correlated randomness [DPSZ12,DSZ15,RWT+18,KVH+21]. However, WW-FL is not bound to any specific MPC setting and each of the MPC clusters as well as the global servers could be instantiated independently using any MPC protocol, as detailed further in §3.3 and §3.5.

$\mathcal{F}_{\textsc{Reshare}}$ Ideal Functionality Given the hybrid nature of WW-FL, the security of the framework heavily relies on the correct instantiation of the ideal functionality for the Reshare algorithm, denoted as $\mathcal{F}_{\textsc{Reshare}}$. This ideal functionality is presented in Alg. 2.

The purpose of $\mathcal{F}_{\textsc{Reshare}}$ is to securely convert secret shares of data D from the sharing semantics of the source U, defined over a party set $\mathcal{P}$ (i.e., $\langle D \rangle_U$) to those of the target V,

Algorithm 2 $\mathcal{F}_{\text{Reshare}}$: Ideal Functionality for Reshare

Input: $\mathcal{P} : \langle D \rangle_U$, $\mathcal{P}' : \perp$
Output: $\mathcal{P} : \perp$, $\mathcal{P}' : \langle D \rangle_V$

1: **secret reconstruction:** $\mathcal{F}_{\text{Reshare}}$ receives $\langle D \rangle_U$ from $\mathcal{P}$ and reconstructs secret D using the sharing semantics of source U.
2: **share generation:** $\mathcal{F}_{\text{Reshare}}$ computes $\langle D \rangle_V$ from D using sharing semantics of target V.
3: **share distribution:** $\mathcal{F}_{\text{Reshare}}$ sends the secret-shares of $\langle D \rangle_V$ to parties in $\mathcal{P}'$ in accordance with the sharing semantics of target V.

defined over a potentially different party set $\mathcal{P}'$ (i.e., $\langle D \rangle_V$). The security guarantees of the Reshare algorithm are inherently tied to the specific instantiation of $\mathcal{F}_{\text{Reshare}}$, which in turn depends on the threat models associated with both U and V. In practice, these instantiations vary based on the deployment setting. Detailed constructions and examples for clusters of 2, 3, and 4 parties are discussed in §3.6.

3.3 Threat Model

Given the hybrid nature of our framework, we adopt a *mixed* adversarial corruption model [BJMS20, HM20] in WW-FL, assuming a centralized adversary $\mathcal{A}$ that orchestrates the corruption[3]. This setup allows the nature of corruption to vary across different MPC clusters, including the global server cluster. We formalize this via a *corruption strategy* St_δ defined over each entity set $\delta \in \{\mathcal{G}, \mathcal{M}_i, \mathcal{C}_i\}_{i \in \mathsf{N}_\mathcal{M}}$.

For instance, $\mathcal{A}$ may choose to maliciously corrupt certain MPC clusters $\mathcal{M}_i$ (reflecting an unjustified trust assumption in the cluster), while corrupting others, including the global servers $\mathcal{G}$, in a semi-honest manner. Within each entity set, $\mathcal{A}$ may corrupt a minority or a majority of parties, depending on the number of honest parties assumed (with the minimal assumption that each set contains at least one honest entity).

To reflect the hybrid structure of WW-FL composed of multiple MPC clusters, we constrain the adversary $\mathcal{A}$ to employ a *uniform corruption type* within each cluster – either semi-honest or malicious – but not a mixture of both. Allowing mixed corruption types within a single cluster [Can00, HMZ08], while a natural extension, is left as future work.

Table 4: Possible configurations of WW-FL in terms of security against various corruption strategies of the adversary. Here $0 < t < \mathsf{N}_\mathcal{M}$.

Security	Global: $\mathcal{G}$	Clusters: $\mathcal{M}$	Clients: $\mathcal{C}$
I-A	semi-honest	$\mathsf{N}_\mathcal{M}$ semi-honest	semi-honest
I-B	semi-honest	t malicious	semi-honest
I-C	semi-honest	$\mathsf{N}_\mathcal{M}$ malicious	semi-honest
II-A	malicious	$\mathsf{N}_\mathcal{M}$ semi-honest	semi-honest
II-B	malicious	t malicious	semi-honest
II-C	malicious	$\mathsf{N}_\mathcal{M}$ malicious	semi-honest
III-A	semi-honest	$\mathsf{N}_\mathcal{M}$ semi-honest	malicious
III-B	semi-honest	t malicious	malicious
III-C	semi-honest	$\mathsf{N}_\mathcal{M}$ malicious	malicious
IV-A	malicious	$\mathsf{N}_\mathcal{M}$ semi-honest	malicious
IV-B	malicious	t malicious	malicious
IV-C	malicious	$\mathsf{N}_\mathcal{M}$ malicious	malicious

The resulting security spectrum of WW-FL is summarized in Tab. 4, ranging from the weaker semi-honest cases to the strongest all-malicious setting, depending on the

[3] This is a stronger notion than having multiple independent adversaries.

adversary's corruption strategy across entities.

$\mathcal{F}_{\textsf{WW-FL}}$ Ideal Functionality We model the FL training process in WW-FL as an ideal functionality $\mathcal{F}_{\textsf{WW-FL}}$ which is depicted in Alg. 3. For simplicity, we assume that the clients have secret-shared their training data with their respective MPC clusters via the respective Share protocol. In each iteration t, $\mathcal{F}_{\textsf{WW-FL}}$ receives the shares of the current model W_{t-1} from the global servers ($\mathcal{G}$) and the shares of the training data from each cluster's servers in $\mathcal{M}_i$ for $i \in \mathsf{N}_{\mathcal{M}}$. Using these shares, $\mathcal{F}_{\textsf{WW-FL}}$ reconstructs the current global model and the training data for each cluster. Next, $\mathcal{F}_{\textsf{WW-FL}}$ trains the model W_{t-1} using the data from each cluster, resulting in a cluster model. These cluster models are then combined using an aggregation algorithm to obtain the updated global model W_t for this iteration. Finally, $\mathcal{F}_{\textsf{WW-FL}}$ generates the secret-shares of W_t among the global servers ($\mathcal{G}$) and the cluster servers ($\mathcal{M}_i$) using their respective sharing semantics.

Algorithm 3 $\mathcal{F}_{\textsf{WW-FL}}$: Ideal Functionality for WW-FL

Actors: $\mathcal{F}_{\textsf{WW-FL}}, \mathcal{G}, \mathcal{M}$ # $\mathcal{M} = \bigcup_i \mathcal{M}_i$; $i \in \mathsf{N}_{\mathcal{M}}$
Input: $\langle W_{t\text{-}1} \rangle$, $\{\langle D_t^{\mathcal{M}_i} \rangle\}_{\mathcal{M}_i \in \mathcal{M}}$ # $W_{t\text{-}1}$ – current global model, $D_t^{\mathcal{M}_i}$ – data at $\mathcal{M}_i$
Output: $\langle W_t \rangle_{\mathcal{G}}$, $\langle W_t \rangle_{\mathcal{M}_i}$ # W_t – global model after iteration t

initialize: $\mathcal{F}_{\textsf{WW-FL}}$ receives $\langle W_{t\text{-}1} \rangle$ from $\mathcal{G}$ and reconstructs $W_{t\text{-}1}$.
for each MPC cluster $\mathcal{M}_i \in \mathcal{M}$ **do** # in parallel
  data reconstruction: $\mathcal{F}_{\textsf{WW-FL}}$ receives $\langle D_t^{\mathcal{M}_i} \rangle$ from $\mathcal{M}_i$ and reconstructs the cluster data $D_t^{\mathcal{M}_i}$. # $D_t^{\mathcal{M}_i}$ corresponds to collective data at $\mathcal{M}_i$
  training: $\mathcal{F}_{\textsf{WW-FL}}$ trains the current global model $W_{t\text{-}1}$ on $D_t^{\mathcal{M}_i}$ to obtain the cluster model $W_t^{\mathcal{M}_i}$.
end for
aggregation: $\mathcal{F}_{\textsf{WW-FL}}$ aggregates $\{W_t^{\mathcal{M}_i}\}_{i \in \mathsf{N}_{\mathcal{M}}}$ using the respective secure aggregation algorithm to obtain W_t.
secret-sharing (global servers): $\mathcal{F}_{\textsf{WW-FL}}$ generates $\langle W_t \rangle_{\mathcal{G}}$ from W_t.
secret-sharing (cluster servers): $\mathcal{F}_{\textsf{WW-FL}}$ generates $\langle W_t \rangle_{\mathcal{M}_i}$ from W_t for $i \in \mathsf{N}_{\mathcal{M}}$.

Security of WW-FL We adopt the standard stand-alone model of [Can00], assuming a static adversary and synchronous communication over perfectly secure channels[4]. Our security analysis is carried out in a simulation-based framework [Lin17], which offers a modular and composable approach to proving security – particularly well-suited to the clustered architecture of the WW-FL framework.

Specifically, for most subprotocols, we first define the corresponding *ideal functionalities*. These functionalities formalize their expected behavior in an ideal execution with a trusted third party, as is standard in MPC. Security of the overall framework is then established via a hybrid proof: each subprotocol securely realizes its functionality, and the composition theorem allows us to combine these results without re-deriving the security of every underlying instantiation [Lin17]. Importantly, we do not assume the existence of corresponding "ideal implementations". Instead, we instantiate the functionalities using existing MPC protocols with proven security guarantees, and compose them within WW-FL to implement more complex tasks.

Let $\mathcal{F}^{\gamma}_{\textsf{MPC}}$ denote the set of MPC ideal functionalities associated with the entity set $\gamma \in \{\mathcal{G}, \mathcal{M}_i\}_{i \in \mathsf{N}_{\mathcal{M}}}$. These functionalities vary depending on the role of the entity in the WW-FL framework. Specifically, for a standard FL setup,[5]

$$\mathcal{F}^{\gamma}_{\textsf{MPC}} = \begin{cases} \{\mathcal{F}^{\mathcal{M}_i}_{\textsc{Share}}, \mathcal{F}^{\mathcal{M}_i}_{\textsc{Reshare}}, \mathcal{F}^{\mathcal{M}_i}_{\textsc{Sample}}, \mathcal{F}^{\mathcal{M}_i}_{\textsc{Train}}\} & \text{if } \gamma = \mathcal{M}_i, \\ \{\mathcal{F}^{\mathcal{G}}_{\textsc{Share}}, \mathcal{F}^{\mathcal{G}}_{\textsc{Reshare}}, \mathcal{F}^{\mathcal{G}}_{\textsc{Agg}}\} & \text{if } \gamma = \mathcal{G}. \end{cases}$$

[4] In practice, such communication channels can be instantiated via mutually authenticated TLS. [5] In the presence of defense mechanisms, $\mathcal{F}^{\gamma}_{\textsf{MPC}}$ may include additional functionalities as listed in Tab. 3; these are straightforward generalizations.

Given these definitions, the overall security of the WW-FL framework is formally stated in Theorem 1.

Theorem 1. *Let $\mathcal{F}^{\gamma}_{\mathsf{MPC}}$ denote the set of ideal MPC functionalities for each entity set $\gamma \in \{\mathcal{G}, \mathcal{M}_i\}_{i \in \mathsf{N}_{\mathcal{M}}}$. Algorithm* FL-Train *(Alg. 1) securely realizes the $\mathcal{F}_{\mathsf{WW\text{-}FL}}$ functionality (Alg. 3) in the $\mathcal{F}^{\gamma}_{\mathsf{MPC}}$-hybrid model, assuming a static adversary $\mathcal{A}$ who corrupts the entities in $\delta \in \{\mathcal{G}, \mathcal{M}_i, \mathcal{C}_i\}_{i \in \mathsf{N}_{\mathcal{M}}}$, according to a predefined corruption strategy St_{δ}.*

Proof Sketch The FL-Train algorithm (Alg. 1) begins with the generation of secret shares of the initial global model W_0 among the global servers $\mathcal{G}$. This operation is performed by the designated stakeholder responsible for deploying the WW-FL framework, using the Share algorithm instantiated for $\mathcal{G}$. The security of this sharing procedure is guaranteed by the ideal functionality $\mathcal{F}^{\mathcal{G}}_{\text{Share}}$.

In an iteration $t \in [1, T]$, the global model W_{t-1}, that is secret-shared among the global servers must be securely redistributed to the servers within each Layer II cluster. This redistribution is carried out via the Reshare protocol executed by $\mathcal{G}$ with respect to each cluster $\mathcal{M}_i$, for all $i \in \mathsf{N}_{\mathcal{M}}$. The security of this step follows from the ideal functionality $\mathcal{F}^{\mathcal{G}}_{\text{Reshare}}$ (cf. Alg. 2), where the source party set is $\mathcal{P} = \mathcal{G}$, holding the secret-shared model $\langle W_{t-1} \rangle_{\mathcal{G}}$, and the target party set is $\mathcal{P}' = \mathcal{M}_i$, which receives the reshared model $\langle W_{t-1} \rangle_{\mathcal{M}_i}$.

In the next step of the FL-Train algorithm, the selected clients C_i^j associated with cluster $\mathcal{M}_i$ (i.e., those in the set $\mathcal{C}_i^{\mathsf{U}}$ securely sampled via the Sample algorithm) generate secret shares of their local data $D_t^{\mathsf{C}_i^j}$ among the servers in $\mathcal{M}_i$, denoted by $\langle D_t^{\mathsf{C}_i^j} \rangle_{\mathcal{M}_i}$. This sharing is performed using the Share algorithm instantiated for $\mathcal{M}_i$, and its security is ensured by the corresponding ideal functionality $\mathcal{F}^{\mathcal{M}_i}_{\text{Share}}$. Subsequently, each cluster $\mathcal{M}_i$ performs ML training on its collective dataset $\langle D_t \rangle_{\mathcal{M}_i}$ using the Train algorithm, whose security is guaranteed by the corresponding ideal functionality $\mathcal{F}^{\mathcal{M}_i}_{\text{Train}}$.

Even in the presence of malicious corruption, the ideal functionality $\mathcal{F}^{\mathcal{M}_i}_{\text{Share}}$ guarantees that a malicious client cannot introduce inconsistent shares across the servers in $\mathcal{M}_i$. While a malicious client may tamper with its input data prior to the secret-sharing process, this behavior falls outside the scope of MPC itself and is typically addressed through complementary defense mechanisms (cf. §5 for defenses in WW-FL). As such, $\mathcal{F}^{\mathcal{M}_i}_{\text{Train}}$ ensures the privacy and correctness of the ML training on data that has been secret-shared by clients. In cases where integrity violations arise due to adversarial behavior, the underlying secure aggregation protocol Agg, which realizes the ideal functionality $\mathcal{F}^{\mathcal{G}}_{\text{Agg}}$, plays a critical role in detecting and mitigating such attacks.

In the next step, the local model $\langle W_t^i \rangle_{\mathcal{M}_i}$ is securely reshared with the global servers $\mathcal{G}$ using the Reshare protocol. The security of this resharing step is ensured by the ideal functionality $\mathcal{F}^{\mathcal{M}_i}_{\text{Reshare}}$, where the source party set is $\mathcal{P} = \mathcal{M}_i$, holding the secret-shared cluster model $\langle W_t^i \rangle_{\mathcal{M}_i}$, and the target party set is $\mathcal{P}' = \mathcal{G}$, which receives the reshared model $\langle W_t^i \rangle_{\mathcal{G}}$.

Finally, the global servers $\mathcal{G}$ execute the Agg algorithm to securely aggregate the models received from all clusters, i.e., $\{\langle W_t^i \rangle_{\mathcal{G}}\}_{i \in \mathsf{N}_{\mathcal{M}}}$, and compute the updated global model in secret-shared form, denoted by $\langle W_t \rangle_{\mathcal{G}}$. The security of this aggregation step is guaranteed by the ideal functionality $\mathcal{F}^{\mathcal{G}}_{\text{Agg}}$.

Therefore, assuming that all underlying functionalities are securely instantiated in accordance with the specified security assumptions for each set $\delta \in \{\mathcal{G}, \mathcal{M}_i, \mathcal{C}_i\}_{i \in \mathsf{N}_{\mathcal{M}}}$ in WW-FL, the FL-Train algorithm (Alg. 1) securely realizes the ideal functionality $\mathcal{F}_{\mathsf{WW\text{-}FL}}$ (Alg. 3). $\square$

3.4 Private Inference in WW-FL

In WW-FL, after the defined number of training iterations T is completed, the MPC clusters serve as clusters for ML inference. Here, we again utilize PPML techniques to enable clients to privately query their clusters [CRS20, KVH+21, MWCB23, NC23, HSW+25, SBBE25]. Consider the scenario where client C holding query Q wants to use the inference service on a model W that is secret-shared with a cluster $\mathcal{M}_k$. This is accomplished by C generating $\langle Q \rangle_{\mathcal{M}_k}$ using the SHARE algorithm instantiated for cluster $\mathcal{M}_k$. Subsequently, the cluster servers in $\mathcal{M}_k$ running PREDICT on $\langle W \rangle_{\mathcal{M}_k}$ and $\langle Q \rangle_{\mathcal{M}_k}$ to generate the inference result in secret-shared form. Finally, $\mathcal{M}_k$ reveals the result to C using the REVEAL protocol.

3.5 Abstraction of Existing FL Schemes

The abstraction of WW-FL captures multiple existing FL frameworks, including standard FL with a single aggregator [MMR+17], distributed aggregators [TXLZ25], and hierarchical FL schemes [Yan21], as shown in Tab. 5. This abstraction not only simplifies comparisons but also facilitates advanced hybrid designs, including the integration of Differential Privacy [DP20, OA22].

Table 5: Abstraction of existing FL schemes (cf. Tab. 1) in WW-FL (cf. Fig. 2). $\mathcal{S}$ – aggregation server(s), S.Agg. – secure aggregation, and (S.)Agg. – optional secure aggregation. See Tab. 2 for other notations.

Scheme		Layer I	Layer II	Layer III	Remark
Aggregation (Single $\mathcal{S}$)	N_s	$\mathsf{N}_{\mathcal{G}} = 1$	$\mathsf{N}_{\mathcal{M}_i} = 1$	$\mathsf{N}_{\mathcal{C}_i} = 1$	$\mathcal{M}_i = \mathcal{C}_i$
	Role	(S.)Agg.	ML Training		
Aggregation (Multi $\mathcal{S}$)	N_s	$\mathsf{N}_{\mathcal{G}} > 1$	$\mathsf{N}_{\mathcal{M}_i} = 1$	$\mathsf{N}_{\mathcal{C}_i} = 1$	$\mathcal{M}_i = \mathcal{C}_i$
	Role	S.Agg.	ML Training		
Hierarchical FL	N_s	$\mathsf{N}_{\mathcal{G}} = 1$	$\mathsf{N}_{\mathcal{M}_i} = 1$	$\mathsf{N}_{\mathcal{C}_i} > 1$	$\mathcal{M}_i \neq \mathcal{C}_i$
	Role	(S.)Agg.	(S.)Agg.	ML Training	
WW-FL **This Work**	N_s	$\mathsf{N}_{\mathcal{G}} > 1$	$\mathsf{N}_{\mathcal{M}_i} > 1$	$\mathsf{N}_{\mathcal{C}_i} > 1$	$\mathcal{M}_i \neq \mathcal{C}_i$
	Role	S.Agg.	PPML Training	Data Sharing	

Standard FL with a single aggregator (SINGLE $\mathcal{S}$) [MMR+17] is a variant of WW-FL, where each Layer III cluster $\mathcal{C}_i$ consists of only one client that also serves as the MPC cluster server $\mathcal{M}_i$ in Layer II. Thus, it is sufficient to conduct ML training without global model privacy concerns and then send the results to a single global server $\mathcal{G}$ in Layer I for aggregation. The case of distributed aggregators (MULTI $\mathcal{S}$) [TXLZ25] follows similarly, except that secure aggregation is performed at Layer I with multiple ($\mathsf{N}_{\mathcal{G}} > 1$) global servers. Finally, existing hierarchical FL schemes [Yan21] have a similar three-layer architecture as WW-FL, but use a single server at both the global and cluster level ($\mathsf{N}_{\mathcal{G}} = 1$, $\mathsf{N}_{\mathcal{M}_i} = 1$). While WW-FL employs PPML training at the cluster-server level, hierarchical FL uses secure aggregation. Additionally, clients in the hierarchical FL approach perform local model training, as opposed to only data sharing in WW-FL.

Moreover, advanced hybrid designs in the WW-FL framework can accommodate *privileged* clients—such as trusted consortium members—who may contribute data without requiring the global model to remain hidden. In such cases, the client can train the model locally and secret-share the resulting gradients directly with the global aggregation servers $\mathcal{G}$.

Another hybrid could involve collaborative training frameworks like SafeNet [CJO22], which offer robustness against poisoning attacks. SafeNet constructs the global model as an ensemble of individually verified models, using majority voting to determine the final prediction. In the context of WW-FL, this setup can serve as a substitute for a conventional MPC cluster and its associated clients. The ensemble models can then be either aggregated using standard techniques or clustered via MPC-based methods [NRC+22] to obtain a single representative model. From this point, the remaining operations conform to the WW-FL architecture, replacing SafeNet's plaintext operations with their MPC counterparts. Notably, training performed within these MPC clusters—which inherently resist poisoning—can be classified as *attested training*. The global servers may then employ lightweight or mildly robust aggregation mechanisms to incorporate updates from these trusted clusters, thereby reducing the risk of error propagation to higher layers.

While our work primarily focuses on MPC for WW-FL components, we emphasize that our abstraction is general enough to accommodate other privacy-preserving techniques, including those based on homomorphic encryption (HE). For example, as discussed in §2, existing multi-party HE (MHE) schemes preserve global model privacy but face scalability limitations [FTP+21, SPT+21, XHX+23]. Within WW-FL, such an MHE-based setup can be treated as a single Layer II cluster. This allows the framework to leverage additional training data contributed by the MHE setup, instead of excluding it entirely. The main challenge in this integration lies in the Reshare algorithm, which would require converting between HE ciphertexts and MPC-style secret shares. Techniques proposed in works such as MP2ML [BCD+20] and Cheetah [HLHD22] can potentially be adapted to support these conversions, enabling seamless interoperability within WW-FL. We leave the integration of such HE-based clusters into WW-FL as an interesting direction for future work.

3.6 Example Configuration and Workflow in WW-FL

In this section, we outline the processing of the machine learning model within our WW-FL framework (cf. Fig. 2), using one specific configuration as an example from the various options listed in Tab. 4. This configuration, detailed in Tab. 6, involves three MPC clusters in Layer II, each using a different MPC protocol and global servers using the 2PC + helper protocol in CrypTen [KVH+21]. We use $\langle x \rangle_\gamma$ to denote the secret shares of the value x held by all the parties in the set γ.

The discussion follows the WW-FL training algorithm, detailed in Alg. 1 (FL-Train). For simplicity, we omit the process of clients secret-sharing their training datasets with the Layer II MPC clusters, as this can be done naively through the input-sharing protocol Share (see Tab. 3) of the underlying MPC protocol. Instead, we focus on the processing of the ML model among the MPC servers.

To enhance clarity in our discussion, we have divided the FL-Train algorithm into five steps, presented in the order of execution. We describe the workflow throughout these steps, and for reference, we have included the corresponding line numbers from Alg. 1 in the format L123. For the remainder of this section, we assume that all operations are performed over a finite ℓ-bit ring, as required by the underlying MPC protocols. Additionally, to simplify the presentation, we omit the discussion of communication optimizations involving shared-key setups used in these protocols, and we refer the reader to [KVH+21, DSZ15, AFL+16, MR18, KPRS22] for further details.

3.6.1 Step I (L1): Initialization of Global Model

In this step, the initial global model W_0, likely owned by an organization or corporate entity, is secret-shared among the two global servers $\mathsf{GS}_1, \mathsf{GS}_2 \in \mathcal{G}$. The model owner uses CrypTen's [KVH+21] input sharing protocol Share to secret-share W_0, resulting in $\langle W_0 \rangle_{\mathcal{G}}$, i.e., $W_0 = \langle W_0 \rangle_{\mathsf{GS}_1} + \langle W_0 \rangle_{\mathsf{GS}_2}$.

Table 6: Example configuration for WW-FL with three MPC Clusters in Layer II (cf. Fig. 2). $\langle x \rangle_\gamma$ represents the secret-shares of value x held by all the parties in the set γ. In the 4PC setting [KPRS22], λ_x^j denotes the input-independent shares, and m_x denotes the input-dependent share of x.

MPC Servers	MPC Protocol	Details
Layer I Global Servers: $\mathcal{G}$	2PC + Helper [KVH+21]	– semi-honest security. – $x = x_1 + x_2$. – $\langle x \rangle_{\mathcal{G}} : (\mathsf{GS}_1 : x_1, \mathsf{GS}_2 : x_2)$.
Layer II MPC Cluster 1: $\mathcal{M}_1$	2PC Additive [DSZ15]	– semi-honest security. – $x = x_1 + x_2$. – $\langle x \rangle_{\mathcal{M}_1} : \left(\mathsf{CS}_1^1 : x_1, \mathsf{CS}_1^2 : x_2\right)$.
Layer II MPC Cluster 2: $\mathcal{M}_2$	3PC Replicated [AFL+16, MR18]	– semi-honest security. – $x = x_1 + x_2 + x_3$. – $\langle x \rangle_{\mathcal{M}_2}$: $\mathsf{CS}_2^1 : (x_1, x_2)$, $\mathsf{CS}_2^2 : (x_2, x_3)$ $\mathsf{CS}_2^3 : (x_3, x_1)$
Layer II MPC Cluster 3: $\mathcal{M}_3$	4PC Replicated [KPRS22]	– malicious security. – $x = \mathsf{m}_x - \lambda_x^1 - \lambda_x^2 - \lambda_x^3$. – $\langle x \rangle_{\mathcal{M}_3}$: $\mathsf{CS}_3^1 : (\mathsf{m}_x, \lambda_x^1, \lambda_x^2)$, $\mathsf{CS}_3^2 : (\mathsf{m}_x, \lambda_x^2, \lambda_x^3)$, $\mathsf{CS}_3^3 : (\mathsf{m}_x, \lambda_x^3, \lambda_x^1)$, $\mathsf{CS}_3^4 : (\lambda_x^1, \lambda_x^2, \lambda_x^3)$.

3.6.2 Step II (L4): Resharing of Global Model

This step involves *resharing* the global model W_{t-1} for the current iteration t, to the three MPC clusters. At a high level, each global server GS_i acts as an input party for the MPC protocol within cluster $\mathcal{M}_j$ and secret-shares its share of the global model. Specifically, for global model $x = W_{t-1}$, GS_i secret-shares $x_i = \langle W_{t-1} \rangle_{\mathsf{GS}_i}$, among the cluster servers $\mathcal{M}_j$, represented as $\langle x_i \rangle_{\mathcal{M}_j}$.

Due to the linearity of MPC schemes, the cluster servers can locally add these shares. This results in a secret sharing of $x = W_{t-1}$ across the servers in cluster $\mathcal{M}_j$, represented as $\langle x \rangle_{\mathcal{M}_j} = \langle x_1 \rangle_{\mathcal{M}_j} + \langle x_2 \rangle_{\mathcal{M}_j}$.

MPC Cluster $\mathcal{M}_1$ This cluster uses 2-out-of-2 additive secret sharing with semi-honest security [DSZ15]. To generate $\langle x \rangle_{\mathcal{M}_1}$ with $x = W_{t-1}$, the servers perform the following steps:

- Each $\mathsf{GS}_i \in \mathcal{G}$ samples random shares $x_{i,1}$ and $x_{i,2}$ such that $x_i = x_{i,1} + x_{i,2}$.
- Each GS_i sends $x_{i,j}$ to cluster server $\mathsf{CS}_1^j \in \mathcal{M}_1$.
- Each CS_1^j locally computes $\langle x \rangle_{\mathsf{CS}_1^j} = x_{1,j} + x_{2,j}$.

MPC Cluster $\mathcal{M}_2$ This cluster is instantiated with a three-party replicated secret sharing with semi-honest security [AFL+16, MR18]. To generate $\langle x \rangle_{\mathcal{M}_2}$ with $x = W_{t-1}$, the servers proceed as follows:

- Each $\mathsf{GS}_i \in \mathcal{G}$ samples random shares $x_{i,1}$, $x_{i,2}$ and $x_{i,3}$ such that $x_i = x_{i,1}+x_{i,2}+x_{i,3}$.
- GS_i sends the pair $(x_{i,j},\, x_{i,j\%3+1})$ to $\mathsf{CS}_2^j \in \mathcal{M}_2$ as its share in $\langle x_i \rangle_{\mathcal{M}_2}$.
- Servers in $\mathcal{M}_2$ locally compute $\langle x \rangle_{\mathcal{M}_2} = \langle x_1 \rangle_{\mathcal{M}_2} + \langle x_2 \rangle_{\mathcal{M}_2}$. Specifically, each CS_2^j computes its share of $\langle x \rangle_{\mathcal{M}_2}$ as

$$\langle x \rangle_{\mathsf{CS}_2^j} = (x_{1,j} + x_{2,j}, x_{1,j\%3+1} + x_{2,j\%3+1}).$$

MPC Cluster $\mathcal{M}_3$ This cluster consists of four servers and utilizes the four-party (4PC) replicated secret sharing with malicious security from Tetrad [KPRS22]. Note that Tetrad uses function-dependent preprocessing to achieve faster online phase. Consequently, the cluster servers in $\mathcal{M}_3$ generate all the input-independent λ values during the preprocessing phase. Our discussion begins at this point, assuming that the preprocessing by the cluster servers has been completed. From here, the servers proceed as follows:

– Cluster servers send the following λ shares to $\mathsf{GS}_i \in \mathcal{G}$:

$$\mathsf{CS}_3^1, \mathsf{CS}_3^3, \mathsf{CS}_3^4 : \lambda_{x_i}^1$$
$$\mathsf{CS}_3^1, \mathsf{CS}_3^2, \mathsf{CS}_3^4 : \lambda_{x_i}^2$$
$$\mathsf{CS}_3^2, \mathsf{CS}_3^3, \mathsf{CS}_3^4 : \lambda_{x_i}^3$$

– GS_i accepts the λ values that form the majority and computes $\lambda_{x_i} = \lambda_{x_i}^1 + \lambda_{x_i}^2 + \lambda_{x_i}^3$.

– GS_i sends $\mathsf{m}_{x_i} = x_i - \lambda_{x_i}$ to $\mathsf{CS}_3^1, \mathsf{CS}_3^2$, and CS_3^3, thus completing the input sharing to generate $\langle x_i \rangle_{\mathcal{M}_3}$.

– Servers in $\mathcal{M}_3$ locally compute $\langle x \rangle_{\mathcal{M}_3} = \langle x_1 \rangle_{\mathcal{M}_3} + \langle x_2 \rangle_{\mathcal{M}_3}$.

3.6.3 Step III (L10): Cluster Training of Local Model

This step involves MPC-based ML training at the cluster servers, using secret shares of the current global model, represented as $\langle W_{t-1} \rangle_{\mathcal{M}_j}$, along with the data from the clients associated with the cluster in the current iteration, denoted by $\langle D_t \rangle_{\mathcal{M}_i}$. The servers then execute the underlying PPML training algorithm, Train, and obtain secret shares of the updated model, denoted by $\langle W_t \rangle_{\mathcal{M}_j}$.

3.6.4 Step IV (L11): Resharing of Local Model

Once each cluster receives its updated local model for iteration t in secret-shared form $\langle W_t \rangle_{\mathcal{M}_j}$, the next step is to reshare this model among the global servers $\mathcal{G}$. This process is similar to Step II above, with the key difference being that the roles of the source and destination of the secret sharing are now reversed. We detail the steps for each of the three Layer II clusters next.

MPC Cluster $\mathcal{M}_1$ To generate $\langle x \rangle_{\mathcal{G}}$ with $x = W_t^1$, the cluster servers in $\mathcal{M}_1$ perform the following steps:

– Each $\mathsf{CS}_1^i \in \mathcal{M}_1$ samples random shares $x_{i,1}$ and $x_{i,2}$ such that $x_i = x_{i,1} + x_{i,2}$.

– Each CS_1^i sends $x_{i,j}$ to global server $\mathsf{GS}_j \in \mathcal{G}$.

– Each GS_j locally computes $\langle x \rangle_{\mathsf{GS}_j} = x_{1,j} + x_{2,j}$.

MPC Cluster $\mathcal{M}_2$ The resharing process in cluster $\mathcal{M}_2$, which uses a three-party replicated sharing scheme where $x = W_t^2$ is represented as $x = x_1 + x_2 + x_3$, closely resembles that of cluster $\mathcal{M}_1$ with a few minor differences.

First, cluster $\mathcal{M}_2$ involves three servers, CS_2^i, compared to the two servers in $\mathcal{M}_1$. Consequently, each global server GS_j will receive secret shares of three values instead of two. Second, the local computation for a secret-shared value x in $\mathcal{M}_2$ involves adding three shares: $\langle x \rangle_{\mathsf{GS}_j} = x_{1,j} + x_{2,j} + x_{3,j}$, rather than just two as in $\mathcal{M}_1$.

Additionally, because $\mathcal{M}_2$ operates under semi-honest security, an optimization can be applied. Specifically, CS_2^1 can directly secret-share the sum $(x_1 + x_2)$, while CS_2^2 secret-shares x_3, effectively eliminating one instance of secret-sharing.

MPC Cluster $\mathcal{M}_3$ Since this cluster uses a maliciously secure four-party protocol from Tetrad [KPRS22], we cannot directly follow the approach where each cluster server CS_3^i secret-shares its share of $x = W_t^3$ with the global servers in $\mathcal{G}$. However, because the protocol uses replicated secret sharing, each share is held by three of the four parties, with at most one being corrupt. Therefore, instead of a single cluster server sharing its own secret, the share can be "jointly" shared by three cluster servers. This ensures that a majority of correct values is always available at the global server GS_j. The remaining steps are similar to those for other clusters and are described below.

– Cluster servers "jointly" secret share the following shares among the two global servers in $\mathcal{G}$:

$$\begin{aligned}
\mathsf{CS}_3^1, \mathsf{CS}_3^3, \mathsf{CS}_3^4 &: \lambda_x^1\\
\mathsf{CS}_3^1, \mathsf{CS}_3^2, \mathsf{CS}_3^4 &: \lambda_x^2\\
\mathsf{CS}_3^2, \mathsf{CS}_3^3, \mathsf{CS}_3^4 &: \lambda_x^3\\
\mathsf{CS}_3^1, \mathsf{CS}_3^2, \mathsf{CS}_3^3 &: \mathsf{m}_x
\end{aligned}$$

– For each of the four instances, GS_i accepts the share that forms the majority.

– Servers in $\mathcal{G}$ locally compute $\langle x\rangle_{\mathcal{G}} = \langle \mathsf{m}_x\rangle_{\mathcal{G}} - \langle \lambda_x^1\rangle_{\mathcal{G}} - \langle \lambda_x^2\rangle_{\mathcal{G}} - \langle \lambda_x^3\rangle_{\mathcal{G}}$.

3.6.5 Step V (L13): Secure Aggregation

In this step, the global servers perform secure aggregation Agg using the cluster models that are secret-shared among them. In our example, these secret shares are $\{\langle W_t^1\rangle_{\mathcal{G}}, \langle W_t^2\rangle_{\mathcal{G}}, \langle W_t^3\rangle_{\mathcal{G}}\}$, as obtained in Step IV. After the Agg protocol, the global servers $\mathcal{G}$ receive secret shares of the updated global model for iteration t, denoted by $\langle W_t\rangle_{\mathcal{G}}$. This updated model will serve as the starting point for iteration $t+1$.

In this example, the global servers are implemented using CrypTen [KVH+21], and the secure aggregation can be executed using the WW-FL code, which provides an implementation of secure aggregation in CrypTen. However, the global servers are not limited to CrypTen. They can also be instantiated with other MPC protocols. For example, SafeFL [GMS+23] offers a PyTorch connector plugin for using PyTorch with the MP-SPDZ [Kel20] framework, which supports a range of efficient MPC protocols.

Similarly, while we use three distinct MPC protocols with a small number of parties to instantiate the MPC clusters in Layer II, our framework is not limited to these specific protocols. The technique of resharing secret shares is well-known in the MPC literature and has been used in several works such as [AAG+25, BCD+20], though it is less common in the context of federated learning. For example, the "converter unit" in FANNG-MPC [AAG+25] introduces methods to convert secret shares from one set of n parties in a dishonest majority setting with malicious security to another set of $m \neq n$ parties in the same corruption setting. Similarly, MP2ML [BCD+20] presents a scheme for converting between HE (using the CKKS scheme) and MPC to enable private ML inference. These works highlight the flexibility of our WW-FL framework, which enables global federated learning and integrates various existing techniques.

4 Performance Evaluation

We systematically and empirically evaluate the performance of WW-FL in terms of accuracy as well as computation and communication overhead. We use standard image classification tasks and build a prototype implementation[6] using the MPC framework

[6] https://encrypto.de/code/WW-FL

CrypTen [KVH+21]. Future research can extend our evaluation to stronger security models (cf. §3.3), as well as more sophisticated ML models and training tasks.

4.1 Implementation

We implement WW-FL based on the CrypTen MPC framework developed by Meta [KVH+21]. CrypTen provides a PyTorch-style interface and implements MPC operations with GPU support. Specifically, it implements semi-honest arithmetic and Boolean secure two- and multi-party computation protocols that use a trusted third "helper" party to generate correlated randomness. CrypTen provides a "simulation" mode where the specified computation is performed on a single node in plaintext yet simulates all effects that computation in MPC would have on the results (e.g., due to limited fixed-point precision and truncation). We leverage this mode to efficiently evaluate WW-FL's accuracy and later the impact of data-poisoning attacks. For our run-time and communication measurements, we benchmark the full MPC protocols. In all our experiments, the fixed-point precision in CrypTen is set to 22 decimal bits (the maximum value recommended by the developers).

We use CrypTen to implement (a) private training on Layer II and (b) distributed aggregation on Layer I. CrypTen out of the box supports private inference between Layer II and III. We extend CrypTen with an identity layer to enable model conversions and re-sharing. Additionally, we extend the implementation of convolutional layers to enable full GPU-accelerated training for such model architectures. Moreover, we provide the necessary code to orchestrate the various parties and components, thereby creating a unified simulation framework.

Moving our prototype implementation to production requires hardening both the cryptographic and system layers. For example, all processing of sensitive data (even when secret-shared) must happen in constant time to protect against timing-based side-channel attacks. Network connections must be secured, e.g., via mutually authenticated TLS to instantiate communication channels that are assumed to be perfectly secure. Finally, servers must be hardened, their APIs must be secured, and reliable key and identity management must be implemented. Please refer to recommended practice for MPC-based systems defined by ITU-T X.1770 [ITU21] and IEEE P2842 [IEE21] for details.

Setup Plaintext FL and CrypTen-based WW-FL *simulations* are run on a single computing platform with two Intel Xeon Platinum 8168 CPUs, 1.5TB RAM, and 16 NVIDIA Tesla V100 GPUs. For realistic results in a *distributed MPC deployment* with two computational and one helper party, we use three Amazon AWS g3s.xlarge instances with 4 vCPUs, and 8GB GPU memory on an NVIDIA Tesla M60. These instances are located in the same AWS availability zone (due to high costs associated with routing traffic between different zones), yet we simulate intra- and inter-continental network connections by setting the bandwidth to 1Gbps/100Mbps and latency to 20ms/100ms respectively.

Tasks Following prior work such as [RSWP23, BMP+24, TXLZ25], we evaluate WW-FL on two standard image classification tasks: recognizing (a) hand-written digits using LeNet [AAG+25] trained on MNIST [LBBH98] dataset, and (b) objects in one of 10 classes using ResNet9 trained on CIFAR10 [Kri09] dataset. We simulate 1000 clients overall from which 100 are randomly sampled during each training iteration. In the FL setting, all sampled clients provide their computed update directly to the aggregation server. In the WW-FL setting, each of the 10 MPC clusters has 100 associated clients from which each samples 10 clients at random. Each client has 200 data points assigned to it at random with duplicates allowed between clients.

For plain FL, we use batch size 8 and learning rate 0.005, and train locally for 5 epochs before central aggregation. To allow for a fair comparison between FL and WW-FL, we use a correspondingly *scaled* batch size of 80 and a learning rate of 0.05 as in [GDG+17]. Each client and MPC cluster performs 5 local training epochs with either a batch size of 8 or 80, respectively, following the recommendation in [GDG+17]. Tab. 7 summarises all the (hyper) parameters used during the training.

Table 7: Training parameters used in FL and WW-FL.

Parameter	FL	WW-FL
# clients	1000	
# clients selected per round	100	
# MPC clusters	-	10
# clients per MPC cluster	-	100
# clients per MPC cluster per round	-	10
size of client datasets	200	
learn rate	0.005	0.05
# local epochs	5	
batch size	8	80

4.2 Results

In this section, we answer the following research questions to empirically evaluate performance and discuss our results:

Q1: What is the accuracy difference between FL [MMR+17] and WW-FL (in plaintext)?

Q2: What is the impact on accuracy for WW-FL when moving from plaintext to MPC?

Q3: What are the run-time and communication overheads of (MPC-based) WW-FL compared to FL?

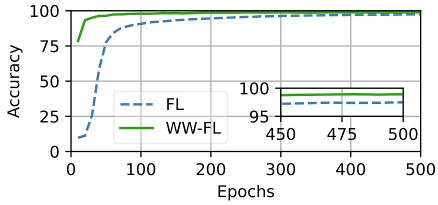
Q4: How does WW-FL scale and compare to existing fully private training approaches?

Q1 – FL vs WW-FL In Fig. 3, we compare the validation accuracy of FL and WW-FL for image classification tasks for 500 and 2000 rounds. Here, WW-FL converges significantly faster than regular FL, e.g., after 500 rounds of training ResNet9 on CIFAR10, WW-FL reaches 85.68% validation accuracy, whereas regular FL only reaches 65.95%. Similarly, for MNIST, WW-FL achieves 98.95% accuracy compared to 98.72% for regular FL. We attribute this to WW-FL pooling training data at the cluster level and thus being able to exploit the known benefits of batching [GDG+17, BCN18, WWCJ22].

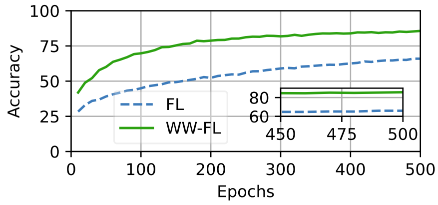
Q2 – Accuracy Impact of MPC In Fig. 4, we compare the plaintext validation accuracy (cf. Q1) to our CrypTen simulation to measure the impact of MPC (i.e., fixed-point representation with 22 bit decimal representation and truncation). While there is a slight difference in initial rounds, both implementations quickly converge to almost the same validation accuracy, with only a small difference on the order of 0.1%.

Q3 – MPC Overhead Finally, we evaluate the overhead of MPC for secure training and aggregation. For this, we measure the run-times and communication for one iteration of LeNet/MNIST training (i.e., 5 local epochs) in AWS for one cluster (with 1Gbps bandwidth and 20ms latency) and one iteration of global aggregation (with 100Mbps bandwidth and 100ms latency). The training for WW-FL on the cluster level takes 315s and requires 5.28GB inter-server communication, which is multiple orders of magnitude more than local plaintext training in PyTorch (which only takes 0.07s). The aggregation over 10 cluster inputs is very efficient with 0.023s run-time and has no communication overhead since only linear operations are required, which are local operations in MPC.

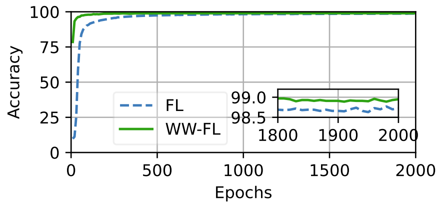
Additional overhead that must be considered for clients is sharing data with the training cluster servers. In our setup, clients on expectation have to upload 3.31MB and 9.86MB in total for 500 rounds of training for MNIST and CIFAR10, respectively. Furthermore, we have to account for sharing the trained models from training clusters to the aggregation



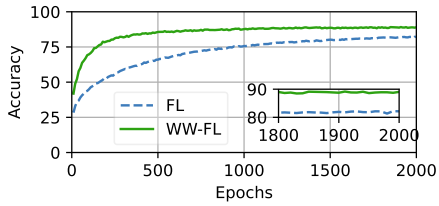
(a) LeNet trained on MNIST for 500 epochs.



(b) ResNet9 on CIFAR10 for 500 epochs.



(c) LeNet trained on MNIST for 2000 epochs.



(d) ResNet9 on CIFAR10 for 2000 epochs.

Figure 3: Validation accuracy for FL and WW-FL.

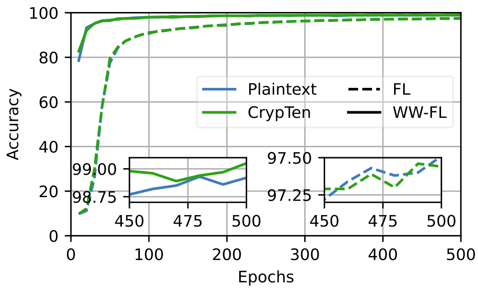


Figure 4: Validation accuracy for FL and WW-FL in plaintext and MPC (CrypTen simulation) for LeNet/MNIST training.

servers. Given the number of model parameters and CrypTen sharing semantics, each training cluster must transfer 0.49MB and 39.19MB per server for LeNet and ResNet9, respectively. This clearly shows that it is significantly more efficient for participants to upload their training data in secret-shared form compared to down- and uploading model parameters for each training round. In our evaluation setup, the training clusters and the aggregation layer use the same MPC configuration, hence no interactive re-sharing is necessary.

Q4 – Scalability & Comparison The only existing solutions for privacy-preserving training and inference with cryptographic guarantees and global model privacy are based on either MPC or MHE (cf. §2). Existing MPC-based PPML solutions with global model privacy can be seen as a special case of WW-FL with a single MPC cluster on Layer II and no aggregation at Layer I (cf. Tab. 5). In terms of MHE, Spindle [FTP+21], Poseidon [SPT+21], and Hercules [XHX+23] are currently the most promising solutions for collaborative training and inference with global model privacy. We therefore compare scalability and especially training overheads with these existing approaches.

For the comparison with MPC, we first assume there is a fixed threshold on the communication overhead a single MPC training cluster can handle in a single round. For example, we can assume a maximum of ≈5GB of inter-server communication. Based on our experimental results provided for Q3, the maximum number of clients a centralized MPC-based PPML system can handle is ≈10. In contrast, in WW-FL with our example configuration of 10 independent MPC training clusters and one global aggregation cluster, we can easily handle 10× more clients (100 randomly selected per round). Here, the number of supported clients scales linearly in the number of available independent MPC training clusters. *As such, moving from a single MPC training cluster to our WW-FL architecture is warranted whenever the resources of a single cluster are exceeded; this can be as low as 10 clients in our example.* When considering computation overhead, the run-time for a single MPC training cluster (and therefore for a centralized MPC-based PPML system) scales linearly in the number of clients and training samples. In contrast, in WW-FL, through adding new MPC training clusters, additional workload resulting from newly joining clients or an increasing number of training samples can be trivially parallelized, thereby not significantly increasing total training run-time; the only minor increase results from the linearly scaling global aggregation step, which however is at least four orders of magnitude more efficient than training – even with full privacy protection (cf. run-time measurements provided for Q3).

For the comparison with MHE, we refer to the experiments reported in [FTP+21, SPT+21, XHX+23]. SPINDLE's [FTP+21] communication overhead is demonstrated to scale linearly in the number of data providers (Layer III clients in our terminology) and ciphertext size, which in turn depends on the number and dimension of training samples. This is similar to WW-FL, however, we can easily handle additional load from new clients by adding MPC training clusters, thus not increasing the load for existing clusters or clients. Furthermore, SPINDLE only supports linear and logistic regression models, whereas we validate WW-FL with more complex convolutional neural networks.

POSEIDON [SPT+21] extends and improves upon SPINDLE: while it also scales linearly in the number of clients and training samples, it supports various neural network architectures. However, it relies on approximations for non-linear layers due to the restrictions of HE schemes. Specifically, the evaluation includes a 3-layer fully-connected neural network for MNIST classification that achieves 89.9% accuracy after 1000 epochs, whereas we validate WW-FL with a network that includes three convolutional and two fully-connected layers (along with non-linear activation functions and pooling layers) [AAG+25], and surpasses 90% accuracy in less than 100 rounds with 5 local epochs in each round (cf. results reported for Q1). More comparable is the evaluation of [SPT+21] on a model for CIFAR10 classification with two convolution and two fully-connected layers, which we therefore reference in our comparison in Tab. 8. Another notable benchmark of POSEIDON [SPT+21] is training the three-layer SecureML neural network [MZ17] with three clients for 15 epochs on MNIST, which takes 73.1 hours, whereas MPC frameworks such as FALCON [WTB+21] only require 0.56 hours for the same task.

HERCULES [XHX+23] advances the state of the art over POSEIDON by enabling more efficient homomorphic matrix operations together with polynomial approximations for non-linear activation functions. It further demonstrates the feasibility of training with up to 50 clients, achieving secure aggregation without relying on multiple non-colluding servers and providing strong privacy guarantees under the passive security model. These results highlight the promise of multi-party homomorphic encryption (MHE)-based approaches for privacy-preserving federated learning. Nevertheless, MHE-based solutions also bring certain costs: they require a multi-party key setup among all clients, entail sizable ciphertexts that increase storage requirements, and typically only support passive security while lacking dedicated mechanisms against poisoning attacks. For instance, training with 10 clients in HERCULES on MNIST requires a total storage of 49.06GB across the clients. By contrast, WW-FL adopts an orthogonal design point: clients only store their training data in plaintext, while servers hold additive shares of training data and model parameters, resulting in less than 4GB of storage in total. On the downside, however, WW-FL using naive MPC alone requires the availability of at least two non-colluding servers in every

MPC cluster. Moreover, while private training is efficiently parallelized across clusters, the overhead compared to cleartext federated learning becomes significant when considering the overall cost of achieving global model privacy.

In Tab. 8, we summarize performance results for the approaches compared above. For the total run-time and communication overhead for WW-FL, we count data sharing between Layer III clients and Layer II cluster servers, Layer II training, secret-shared parameter up- and download between Layer II and Layer I cluster servers, and secure aggregation on Layer I. For the centralized MPC-based method, we only have to consider data sharing between clients and one MPC cluster with the same configuration as a WW-FL Layer II cluster that executes all training epochs on its own. While this results in slightly reduced communication overhead compared to WW-FL (as secure aggregation can be omitted), there is an order of magnitude increase in run-time as the training procedure cannot be parallelized as for WW-FL. Also, note that it might not be practically feasible for a single cluster to handle the substantial total communication overhead that is evenly distributed among all clusters in WW-FL.

Finally, we report the original benchmark results of Poseidon [SPT+21] and Hercules [XHX+23]. Due to the severely limited scale of experiments in terms of supported number of clients as well as the limited model architecture complexity, a fair comparison to WW-FL is not entirely feasible. However, when relying on the theoretical linear scalability in terms of client numbers and training epochs, we can estimate that Hercules would require roughly 1.86× more run-time overhead than WW-FL for their “CIFAR-10-N1” model that is somewhat comparable yet still less complex than LeNet.

Table 8: Comparison of total run-time in hours, communication in terabytes, and storage in gigabytes for WW-FL and central MPC-based training for 1000 clients, and MHE-based approaches (Poseidon [SPT+21] and Hercules [XHX+23]) for 10 and 50 clients. Note that the neural networks trained for Poseidon and Hercules are less complex and do not include non-linear layers. Type of layers: CV – convolution, MP – max pooling, FC – fully connected, AP – average pooling.

Metric	WW-FL	MPC [KVH+21]	[SPT+21]	[XHX+23]	[SPT+21]	[XHX+23]
# clients	1,000	1,000	10	10	50	50
# clients per round	100	100	10	10	50	50
# epochs in total	2,500	2,500	1,000	1,000	25,000	25,000
Accuracy	98.95%	98.95%	88.7%	91.8%	51.8%	54.3%
Model	LeNet [AAG+25]: 3CV, 4ReLU, 2MP, 2FC		“MNIST” [XHX+23]: 3FC		“CIFAR-10-N1” [XHX+23]: 2CV, 1AP, 1MP, 2FC	
Total run-time (in h)	43.83	437.57	1.24	0.43	126.26	40.73
Total comm. (in TB)	26.45	26.40	7.03	0.18	3,076.17	102.54
Total storage (in GB)	3.39	3.33	3,140.25	49.06	105,468.5	823.50

5 Attacks & Mitigations

In FL, malicious participants can degrade accuracy through data or model poisoning attacks [TCLY22]. Because WW-FL models are not accessible to clients, these attackers are limited to manipulating the training data they provide, resulting in either targeted, backdoor, or untargeted attacks. Our work is focused on untargeted FL poisoning, as it is most relevant to real-world applications and more difficult to detect [FCJG20, SHKR22, RSWP23, TXLZ25]. We therefore propose how to systematically evaluate the effectiveness of state-of-the-art data-poisoning attacks [XXE12, FCJG20, TTGL20, SHKR22] in the WW-FL setting and propose a new robust aggregation scheme as a possible mitigation.

5.1 Data-Poisoning Attacks in WW-FL

In data-poisoning attacks, malicious clients can perform arbitrary manipulations on the training data. State-of-the-art attacks are based on label flipping, where clients keep the legitimate training samples, yet exchange the associated labels according to different strategies [FCJG20, TTGL20, SHKR22].

To evaluate WW-FL, we implemented four label-flipping attacks detailed below. We assume a single attacker controlling all malicious clients, coordinating their actions. Only one attack occurs at a time, and if multiple clients have the same data sample, their poisoned labels will be the same. The assumption is that each malicious client poisons all its data samples to maximize the attack's impact. The data is poisoned once and the labels remain constant throughout the model's training.

- Random Label Flipping (RLF [XXE12]): Each poisoned sample is assigned a random class label, i.e., for $num\_classes$ classes in the dataset, define the new label as $new\_label = \texttt{randint}(0, num\_classes - 1)$.

- Targeted Label Flipping (TLF [TTGL20]): Simply flips all labels from a source class to a target class. In WW-FL, we always set the source class as 0 and the target class as 1.

- Static Label Flipping (SLF [FCJG20,SHKR22]): Uses a fixed permutation to determine the new label for each poisoned sample, given by the equation $new\_label = num\_classes - old\_label - 1$.

- Dynamic Label Flipping (DLF [SHKR22]): Utilizing a surrogate model to flip labels for each sample, our implementation aggregates data from all malicious clients to train a model with the same architecture as the one used in WW-FL training. Once trained, this model is utilized for inference and the labels are assigned to the least probable output determined by the surrogate model. The term "dynamic" is used as the labels rely on the trained model, and by altering the training configuration, the poisoned labels will be modified accordingly. The exact training settings of the surrogate model are given in Tab. 9.

Table 9: Training parameters for surrogate model in DLF.

Parameter	Epochs	Batch Size	Learning Rate	Momentum	Weight Decay
Values	50	128	0.05	0.9	0.000,5

5.2 New Robust Aggregation Scheme

The most common FL aggregation scheme, "FedAvg", simply computes a (weighted) average of all inputs (cf. §2). In contrast, *robust* aggregation schemes detect and exclude outliers, thus a suitable mitigation against data poisoning. From the schemes surveyed in [SHKR22, GMS+23, XFG25], we identify "FLTrust" [CFLG21] and "Trimmed Mean" (TM) [YCRB18] as the ones with most efficient MPC implementations.

FLTrust [CFLG21] measures the cosine similarity between the inputs of participants and the most recent global model trained with a clean data set, then excludes the least similar inputs. Trimmed Mean (TM) [YCRB18], for each coordinate, computes the mean across the provided gradient updates and excludes the values that deviate the most in either direction of the mean. Performing TM aggregation obliviously in MPC requires implementing costly sorting to determine the ranking in each coordinate.

We observe that, intuitively, data poisoning in contrast to model poisoning does not result in specific coordinates producing extreme outliers. Hence, we propose a heuristic "Trimmed Mean Variant" that computes the mean and ranking only for a small *randomly* *sampled subset* of coordinates. Then, during aggregation, it excludes those inputs that occurred the most as outliers in the sample. We detail our algorithm in Alg. 4, where the underlying MPC functionalities TM-LIST and TOPK-HITTER (cf. Tab. 3) are as follows:

Algorithm 4 Our Trimmed Mean (TM) Variant in WW-FL

Input: $\mathcal{W} = \{W^i\}_{i \in [\mathsf{N}_\mathcal{M}]}$, α, β, γ # $\gamma = |W^i|$, α – trim threshold, β – sample size
Output: $\langle W_{AGG} \rangle$ # aggregated model after removing outliers

initialize: $\mathcal{Z} \leftarrow \emptyset$ # set of outliers
$\mathcal{I} \leftarrow$ sample random β indices from $[1, \gamma]$.
$\mathcal{U} \leftarrow$ TM-LIST$(\mathcal{W}^\mathcal{I}, \alpha)$ # $\mathcal{W}^\mathcal{I}$–Truncated $\mathcal{W}$ with only indices in $\mathcal{I}$, TM-LIST performs Trimmed Mean algorithm and returns 2α outlier values (top and bottom α) for each index in $\mathcal{I}$, $|\mathcal{U}| = 2\alpha\beta$
$\mathcal{V} \leftarrow$ TOPK-HITTER$(\mathcal{U}, 2\alpha)$ # returns list of 2α indices that occur most frequently in $\mathcal{U}$
for all $i \in \mathcal{V}$ **do**
  $\mathcal{Z} \leftarrow \mathcal{Z} \bigcup \{W^i\}$
end for
$\langle W_{AGG} \rangle \leftarrow$ AGG$(\mathcal{W} \setminus \mathcal{Z})$

TM-LIST takes as input a set of vectors, say $\mathcal{W}$, consisting of β-sized vectors of the form W_j^i for $i \in [\beta], j \in [\mathsf{N}_\mathcal{W}]$. Moreover, the values in the vector come from a fixed source (the Layer II MPC clusters in our case) and are thus represented as a tuple of the form $W_j^i = (u_i, v_i)_j$. Here, u_i denotes the source ID (MPC cluster in WW-FL) and v_i represents the corresponding value. W.l.o.g., consider the first index position of these vectors ($i = 1$). TM-LIST sorts the list $\{(u, v)_j\}_{j \in [\mathsf{N}_\mathcal{W}]}$ using the value v as the key and selects the IDs (u) associated with the top and bottom α values. Intuitively, the operation results in selecting the MPC clusters whose local updates fall in either the top-α or bottom-α position among all the updates at that index. This procedure is performed in parallel for all β indices and results in a set $\mathcal{U}$ of $2\alpha\beta$ IDs (with duplicates). The TOPK-HITTER functionality, parameterized by Γ, takes this set $\mathcal{U}$ as input and returns a set of Γ values that occur most frequently in $\mathcal{U}$.

5.3 CrypTen Implementation Details for Trimmed Mean

CrypTen lacks a built-in oblivious sorting functionality, so we implement privacy-preserving sorting for so-called CrypTensors. Sorting is necessary to compute trimmed mean and our optimized trimmed mean variant. We minimize the number of comparisons by implementing a bitonic sorting network that generates the necessary comparisons between elements to allow for a non-stable sorting algorithm. For trimmed mean, it is not necessary to preserve the relative order of same valued keys as each of them would have been seen as suspicious anyway. The comparisons are stored in plaintext, as they do not reveal any information about the data. The comparison steps are only computed once and then executed in parallel for each coordinate. For 100 elements, we perform 1077 comparisons and for 10 elements only 31. As the result of each comparison is hidden, we perform the swap operation for each pairs as described in Lst. 1.

```
compare_indices = list(comparision_generator(tensors.size(1)))
for (i, j) in compare_indices:
  b = tensors[:, i].gt(tensors[:, j])
  h = b * tensors[:, i] + (1 - b) * tensors[:, j]
  l = b * tensors[:, j] + (1 - b) * tensors[:, i]
  tensors[:, i] = l
  tensors[:, j] = h
```

Listing 1: CrypTen: Parallel Sorting.

When computing the proposed trimmed mean variant, we are only interested in the indices of the outliers and therefore only perform the swap operations on a list of indices. This is done to minimize the compute operations and thereby reduce time and communication. After identifying which gradients were most often detected as outliers and then computing the set of benign indices, we finally compute the sum over those while preserving privacy. The procedure is shown in Lst. 2.

```
def bincount(self, indices, num_bins):
    indices_list = crypten.cryptensor(torch.arange(0, num_bins))
    counts = crypten.stack([indices for _ in range(num_bins)],
        dim=1).eq(indices_list).sum(dim=0)
    return counts[None, :]

def exclude_topk_indices(self, indices, k):
    sorted_indices = self.sort(indices).reshape(-1)
    n = sorted_indices.shape[0]
    return sorted_indices[:n - k]

benign_indices = self.exclude_topk_indices(
    self.bincount(outlier_indices, num_bins=n),
    k=self.exclude_topk)

all_indices = crypten.cryptensor(torch.arange(0, n).repeat(benign_indices.size(0),
    1)).t()
eq_mask = all_indices.eq(benign_indices)
agg_sum = gradients.matmul(eq_mask).sum(dim=1)

return agg_sum / (n - self.exclude_topk)
```

Listing 2: CrypTen: Sum of Benign Updates.

Tab. 10 shows the parameters used by the aggregation server. The trimmed mean threshold α has been chosen such that it can exclude all malicious updates on either side in the FL setting. The worst-case scenario we consider is a poison rate of 0.2, which is equivalent to approx. 20 malicious clients selected per round.

Table 10: Robust aggregation parameters.

Parameter	FL	WW-FL
Trimmed Mean α	20	2
Trimmed Mean Variant excluded gradients	40	4
FLTrust root data set size	200	

FLTrust [CFLG21] uses a root data set to calculate the server model. To be most effective, the server model must be similar to the benign models and therefore the root dataset must be representative for the whole dataset. Therefore, we sample (like for all clients) 200 data points for that dataset.

5.4 Evaluation

We now empirically evaluate the impact of data-poisoning attacks on WW-FL considering different (robust) aggregation schemes. For this, we implement all four attacks—RLF [XXE12], TLF [TTGL20], SLF [FCJG20,SHKR22], DLF [SHKR22]—in our framework and add CrypTen-based implementations of three robust aggregation schemes to our prototype implementation. Using the setup described in §4, we answer the following questions next:

Q5: What is the impact of data-poisoning attacks on the accuracy of FL and WW-FL using FedAvg and robust aggregation schemes?

Q6: What is the run-time and communication overhead for different robust aggregation schemes in MPC?

Q7: How does our TM variant compare to regular TM w.r.t. accuracy and MPC performance?

Q5 – Attack Impact We consider three poison rates (0.01, 0.1, 0.2) and distributions of attackers: in the *equally-distributed* setting, we assume that on expectation each cluster has the same number of malicious clients; in the *focused* setting, we assume that malicious clients are concentrated on as few clusters as possible while there is still an honest majority in each cluster (a standard assumption in FL); finally, in the unrealistic *cluster-focused* setting, we see what happens if we lift the honest-majority assumption and concentrate all malicious clients in as few clusters as possible. In Fig. 5, we study the effectiveness of the most powerful DLF [SHKR22] attack on both regular FL and WW-FL when training ResNet9 on CIFAR10 in the equally distributed and focused setting. Additional experimental outcomes, including evaluation over the cluster-focused setting, are given in the full version [MSS+25].

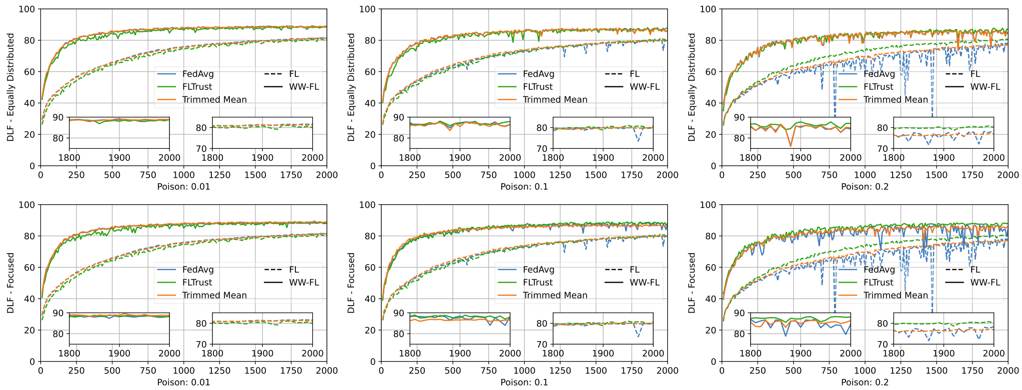


Figure 5: Validation accuracy for FL and WW-FL when training ResNet9 on CIFAR10 with FedAvg, FLTrust, and trimmed mean as aggregation schemes under DLF attack for three different poison rates (top: equally distributed, bottom: focused setting).

For the fairly aggressive poison rate of 0.2, we see in both visualized attacker distributions a significant negative impact of the DLF attack on FL when using FedAvg with drops below 30% accuracy. However, these can be successfully mitigated with robust aggregation schemes. While there is also negative impact on WW-FL, especially in the focused setting, the accuracy even with FedAvg never drops below that of FL. Even though robust aggregation schemes help to slightly smoothen the curve, we conclude that *applying defenses in WW-FL against data-poisoning attacks is optional, but not strictly necessary.*

Q6 – Robust Aggregation in MPC We evaluate the run-time and communication overhead of our FLTrust and TM implementation in CrypTen in Tab. 11. The run-time overhead for both robust aggregation schemes compared to FedAvg is four to five orders of magnitude higher (ranging from 5 to 30 minutes). Also, FLTrust requires 5× more run-time and communication than TM. Given that both produce fairly similar results when applied to WW-FL, the overhead for FLTrust seems not justified.

Table 11: Communication (Comm. in MB) and run-time (Time in seconds) for various aggregation schemes, including our Trimmed Mean (TM) variant with sample sizes 10, 100, and 1000 in CrypTen.

Param	FedAvg	FLTrust	TM	TM-10	TM-100	TM-1000
Comm.	0	5,329.19	1,021.59	9.69	11.90	33.94
Time	0.023	1,713.44	326.57	558.05	558.26	561.50

Q7 – Trimmed Mean Variant In Fig. 6, we compare the effectiveness of our TM variant to the original TM [YCRB18] for three sample sizes (10, 100, and 1000). It turns out that our heuristic approach barely reduces the effectiveness, even with aggressive parameters.

In fact, in the focused setting, the TM variant outperforms the original. This is because our variant completely excludes updates of (poisoned) outliers, whereas in regular trimmed mean, those poisoned updates might still be considered for some coordinates. Results for all other settings are presented in the full version [MSS+25].

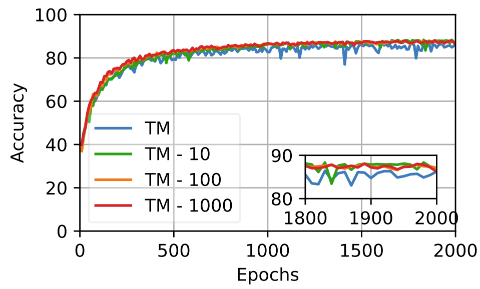


Figure 6: Accuracy of trimmed mean (TM) and our variant (with sample sizes 10, 100, and 1000) against focused DLF attacks on WW-FL at 0.2 poison rate for ResNet9/CIFAR10.

In Tab. 11, we provide run-times and communication results for our optimizations. Compared to the original with 1.02GB of communication, we can see an improvement by two orders of magnitude for the variant with 100 random samples. However, we see a higher and fairly stable run-time across all three examined variants. This is because the algorithm for determining the overall ranking of outliers increases the number of MPC communication rounds. In the studied inter-continental WAN setting, this has severe impact but does not correspond to actual compute time. Overall, if WW-FL is combined with a robust aggregation scheme, our TM variant offers an excellent trade-off between accuracy and MPC overhead.

Additional Benchmarks We provide additional results in the full version [MSS+25], including additional accuracy comparisons of WW-FL with standard FL in all three modes of corruption, and the validation accuracy results for our Trimmed Mean variant.

6 Conclusion

In this work, we presented WW-FL, a novel unified abstraction and framework for large-scale (hierarchical) federated learning that provides global model privacy, faster convergence, smaller attack surface, and better resilience against poisoning attacks than regular FL. We envision that WW-FL will lead to a paradigm shift in FL system design and deployment. Future efforts can extend our foundational work with more efficient instantiations, stronger security, and comprehensive evaluations with larger datasets and setups.

Acknowledgments

This project received funding from the European Research Council (ERC) under the European Union's research and innovation programs Horizon 2020 (PSOTI/850990) and Horizon Europe (PRIVTOOLS/101124778). It was co-funded by the Deutsche Forschungsgemeinschaft (DFG) within SFB 1119 CROSSING/236615297.

References

[AAG+25] Najwa Aaraj, Abdelrahaman Aly, Tim Güneysu, Chiara Marcolla, Johannes Mono, Rogerio Paludo, Iván Santos-González, Mireia Scholz, Eduardo Soria-

Vazquez, Victor Sucasas, and Ajith Suresh. FANNG-MPC: Framework for Artificial Neural Networks and Generic MPC. *IACR CHES*, 2025.

[AAUC18] Abbas Acar, Hidayet Aksu, A. Selcuk Uluagac, and Mauro Conti. A Survey on Homomorphic Encryption Schemes: Theory and Implementation. *ACM Computing Surveys*, 2018.

[AFL+16] Toshinori Araki, Jun Furukawa, Yehuda Lindell, Ariel Nof, and Kazuma Ohara. High-Throughput Semi-Honest Secure Three-Party Computation with an Honest Majority. In *CCS*, 2016.

[BBG+20] James Henry Bell, Kallista A. Bonawitz, Adrià Gascón, Tancrède Lepoint, and Mariana Raykova. Secure Single-Server Aggregation with (Poly)Logarithmic Overhead. In *CCS*, 2020.

[BCD+20] Fabian Boemer, Rosario Cammarota, Daniel Demmler, Thomas Schneider, and Hossein Yalame. MP2ML: A Mixed-Protocol Machine Learning Framework for Private Inference. In *ARES*, 2020.

[BCMC19] Arjun Nitin Bhagoji, Supriyo Chakraborty, Prateek Mittal, and Seraphin B. Calo. Analyzing Federated Learning through an Adversarial Lens. In *ICML*, 2019.

[BCN18] Léon Bottou, Frank E. Curtis, and Jorge Nocedal. Optimization Methods for Large-Scale Machine Learning. *SIAM Review*, 2018.

[BDS+22] Franziska Boenisch, Adam Dziedzic, Roei Schuster, Ali Shahin Shamsabadi, Ilia Shumailov, and Nicolas Papernot. All You Need Is Matplotlib. http://www.cleverhans.io/2022/04/17/fl-privacy.html, 2022.

[BDS+23a] Franziska Boenisch, Adam Dziedzic, Roei Schuster, Ali Shahin Shamsabadi, Ilia Shumailov, and Nicolas Papernot. Reconstructing Individual Data Points in Federated Learning Hardened with Differential Privacy and Secure Aggregation. In *IEEE EuroS&P*, 2023.

[BDS+23b] Franziska Boenisch, Adam Dziedzic, Roei Schuster, Ali Shahin Shamsabadi, Ilia Shumailov, and Nicolas Papernot. When the Curious Abandon Honesty: Federated Learning Is Not Private. In *IEEE EuroS&P*, 2023.

[BEG+19] Kallista A. Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloé Kiddon, Jakub Konečný, Stefano Mazzocchi, Brendan McMahan, Timon Van Overveldt, David Petrou, Daniel Ramage, and Jason Roselander. Towards Federated Learning at Scale: System Design. In *MLSys*, 2019.

[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. In *CCS*, 2017.

[BJMS20] Saikrishna Badrinarayanan, Aayush Jain, Nathan Manohar, and Amit Sahai. Secure MPC: Laziness Leads to GOD. In *ASIACRYPT*, 2020.

[BMGS17] Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. Machine Learning with Adversaries: Byzantine Tolerant Gradient Descent. In *NeurIPS*, 2017.

[BMP+24] Yaniv Ben-Itzhak, Helen Möllering, Benny Pinkas, Thomas Schneider, Ajith Suresh, Oleksandr Tkachenko, Shay Vargaftik, Christian Weinert, Hossein Yalame, and Avishay Yanai. ScionFL: Efficient and Robust Secure Quantized Aggregation. In *IEEE SaTML*, 2024.

[Can00] Ran Canetti. Security and Composition of Multiparty Cryptographic Protocols. *Journal of Cryptology*, pages 143–202, 2000.

[CCPS19] Harsh Chaudhari, Ashish Choudhury, Arpita Patra, and Ajith Suresh. ASTRA: High Throughput 3PC over Rings with Application to Secure Prediction. In *CCSW@CCS*, 2019.

[CFLG21] Xiaoyu Cao, Minghong Fang, Jia Liu, and Neil Zhenqiang Gong. FLTrust: Byzantine-robust Federated Learning via Trust Bootstrapping. In *NDSS*, 2021.

[CJO22] Harsh Chaudhari, Matthew Jagielski, and Alina Oprea. SafeNet: The Unreasonable Effectiveness of Ensembles in Private Collaborative Learning. In *IEEE SaTML*, 2022.

[CK13] Colette Cuijpers and Bert-Jaap Koops. Smart Metering and Privacy in Europe: Lessons from the Dutch Case. In *European Data Protection: Coming of Age*. Springer, 2013.

[CMS+23] Sylvain Chatel, Christian Mouchet, Ali Utkan Sahin, Apostolos Pyrgelis, Carmela Troncoso, and Jean-Pierre Hubaux. PELTA – Shielding Multiparty-FHE against Malicious Adversaries. In *ACM CCS*, 2023.

[Com14] European Commission. Benchmarking Smart Metering Deployment in the EU-27 with a Focus on Electricity. *COM/2014/0356 final*, 2014.

[CRS20] Harsh Chaudhari, Rahul Rachuri, and Ajith Suresh. Trident: Efficient 4PC Framework for Privacy Preserving Machine Learning. In *NDSS*, 2020.

[DCL+21] Ye Dong, Xiaojun Chen, Kaiyun Li, Dakui Wang, and Shuai Zeng. FLOD: Oblivious Defender for Private Byzantine-Robust Federated Learning with Dishonest-Majority. In *ESORICS*, 2021.

[DHW+25] Jiacheng Du, Jiahui Hu, Zhibo Wang, Peng Sun, Neil Zhenqiang Gong, Kui Ren, and Chun Chen. SoK: On Gradient Leakage in Federated Learning. In *USENIX Security*, 2025.

[DLR+21] Yongheng Deng, Feng Lyu, Ju Ren, Yongmin Zhang, Yuezhi Zhou, Yaoxue Zhang, and Yuanyuan Yang. SHARE: Shaping Data Distribution at Edge for Communication-Efficient Hierarchical Federated Learning. In *ICDCS*, 2021.

[DP20] Damien Desfontaines and Balázs Pejó. SoK: Differential Privacies. *PETS*, 2020.

[DPSZ12] Ivan Damgård, Valerio Pastro, Nigel P. Smart, and Sarah Zakarias. Multiparty Computation from Somewhat Homomorphic Encryption. In *CRYPTO*, 2012.

[DR14] Cynthia Dwork and Aaron Roth. The Algorithmic Foundations of Differential Privacy. *Found. Trends Theor. Comput. Sci.*, 2014.

[DSZ15] Daniel Demmler, Thomas Schneider, and Michael Zohner. ABY - A Framework for Efficient Mixed-Protocol Secure Two-Party Computation. In *NDSS*, 2015.

[DXD+23] Chenhe Dong, Yuexiang Xie, Bolin Ding, Ying Shen, and Yaliang Li. Tunable Soft Prompts are Messengers in Federated Learning. In *EMNLP*, 2023.

[EKR18] David Evans, Vladimir Kolesnikov, and Mike Rosulek. A Pragmatic Introduction to Secure Multi-Party Computation. *Foundations and Trends in Privacy and Security*, 2018.

[Eri22] Erin. Data Privacy in Business: A Risk Leading to Major Opportunities. https://matomo.org/blog/2022/08/data-privacy-in-business-risks-and-opportunities/, 2022.

[FCJG20] Minghong Fang, Xiaoyu Cao, Jinyuan Jia, and Neil Zhenqiang Gong. Local Model Poisoning Attacks to Byzantine-Robust Federated Learning. In *USENIX Security*, 2020.

[FGC+22] Liam H. Fowl, Jonas Geiping, Wojciech Czaja, Micah Goldblum, and Tom Goldstein. Robbing the Fed: Directly Obtaining Private Data in Federated Learning with Modified Models. In *ICLR*, 2022.

[FHC+24] Wenzhi Fang, Dong-Jun Han, Evan Chen, Shiqiang Wang, and Christopher G. Brinton. Hierarchical Federated Learning with Multi-Timescale Gradient Correction. In *NeurIPS*, 2024.

[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 *DLS@S&P*, 2021.

[FTP+21] David Froelicher, Juan Ramón Troncoso-Pastoriza, Apostolos Pyrgelis, Sinem Sav, Joao Sa Sousa, Jean-Philippe Bossuat, and Jean-Pierre Hubaux. Scalable Privacy-Preserving Distributed Learning. *PETS*, 2021.

[GBDM20] Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. Inverting Gradients – How Easy Is It to Break Privacy in Federated Learning? In *NeurIPS*, 2020.

[GDG+17] Priya Goyal, Piotr Dollár, Ross B. Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour. *CoRR*, abs/1706.02677, 2017.

[GDL+16] Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, Kristin E. Lauter, Michael Naehrig, and John Wernsing. CryptoNets: Applying Neural Networks to Encrypted Data with High Throughput and Accuracy. In *ICML*, 2016.

[Gen09] Craig Gentry. Fully Homomorphic Encryption Using Ideal Lattices. In *STOC*, 2009.

[GGP24] Rachid Guerraoui, Nirupam Gupta, and Rafael Pinot. Byzantine Machine Learning: A Primer. *ACM Computing Surveys*, 2024.

[GJS+23] Zahra Ghodsi, Mojan Javaheripi, Nojan Sheybani, Xinqiao Zhang, Ke Huang, and Farinaz Koushanfar. zPROBE: Zero Peek Robustness Checks for Federated Learning. In *ICCV*, 2023.

[GMS+23] Till Gehlhar, Felix Marx, Thomas Schneider, Ajith Suresh, Tobias Wehrle, and Hossein Yalame. SAFEFL: MPC-friendly Framework for Private and Robust Federated Learning. In *DLS@S&P*, 2023.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to Play any Mental Game or A Completeness Theorem for Protocols with Honest Majority. In *STOC*, 1987.

[Hib22] Gary Hibberd. The actual cost of a breach – reputation, loss of customers, fines, suspension of business. https://www.tripwire.com/state-of-security/actual-cost-breach-reputation-loss-customers-fines-suspension-business, 2022.

[Hil23] Michael Hill. The biggest data breach fines, penalties, and settlements so far. https://www.csoonline.com/article/567531/the-biggest-data-breach-fines-penalties-and-settlements-so-far.html, 2023.

[HLHD22] Zhicong Huang, Wen-jie Lu, Cheng Hong, and Jiansheng Ding. Cheetah: Lean and Fast Secure Two-Party Deep Neural Network Inference. In *USENIX Security*, 2022.

[HLS+20] Chaoyang He, Songze Li, Jinhyun So, Mi Zhang, Hongyi Wang, Xiaoyang Wang, Praneeth Vepakomma, Abhishek Singh, Hang Qiu, Li Shen, Peilin Zhao, Yan Kang, Yang Liu, Ramesh Raskar, Qiang Yang, Murali Annavaram, and Salman Avestimehr. FedML: A Research Library and Benchmark for Federated Machine Learning. *CoRR*, abs/2007.13518, 2020.

[HM20] Martin Hirt and Marta Mularczyk. Efficient MPC with a Mixed Adversary. In *ITC*, 2020.

[HMZ08] Martin Hirt, Ueli M. Maurer, and Vassilis Zikas. MPC vs. SFE: Unconditional and Computational Security. In *ASIACRYPT*, 2008.

[HSW+25] Christopher Harth-Kitzerow, Ajith Suresh, Yonqing Wang, Hossein Yalame, Georg Carle, and Murali Annavaram. High-Throughput Secure Multiparty Computation with an Honest Majority in Various Network Settings. *PETS*, 2025.

[Hus23] Osman Husain. 51 Biggest GDPR Fines and Penalties So Far. https://www.enzuzo.com/blog/biggest-gdpr-fines, 2023.

[HV19] Cheol-Ho Hong and Blesson Varghese. Resource Management in Fog/Edge Computing: A Survey on Architectures, Infrastructure, and Algorithms. *ACM Computing Surveys*, 2019.

[IEE21] IEEE. P2842: Recommended Practice for Secure Multi-Party Computation. https://ieeexplore.ieee.org/document/9604029, 2021.

[ITU21] ITU. X.1770: Technical Guidelines for Secure Multi-Party Computation. https://www.itu.int/rec/T-REC-X.1770/en, 2021.

[Kel20] Marcel Keller. MP-SPDZ: A Versatile Framework for Multi-Party Computation. In *ACM CCS*. ACM, 2020.

[KMY+16] Jakub Konečný, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated Learning: Strategies for Improving Communication Efficiency. *CoRR*, abs/1610.05492, 2016.

[KPRS22] Nishat Koti, Arpita Patra, Rahul Rachuri, and Ajith Suresh. Tetrad: Actively Secure 4PC for Secure Training and Inference. In *NDSS*, 2022.

[Kri09] Alex Krizhevsky. Learning Multiple Layers of Features from Tiny Images. Technical report, University of Toronto, 2009.

[KS22] Marcel Keller and Ke Sun. Secure Quantized Training for Deep Learning. In *ICML*, 2022.

[KVH+21] Brian Knott, Shobha Venkataraman, Awni Y. Hannun, Shubho Sengupta, Mark Ibrahim, and Laurens van der Maaten. CrypTen: Secure Multi-Party Computation Meets Machine Learning. In *NeurIPS*, 2021.

[LBBH98] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based Learning Applied to Document Recognition. *Proceedings of the IEEE*, 1998.

[LBG+25] Zhiqiang Li, Haiyong Bao, Menghong Guan, Hao Pan, Cheng Huang, and Hong-Ning Dai. EBS-CFL: Efficient and Byzantine-robust Secure Clustered Federated Learning. In *AAAI*, 2025.

[LBT+20] Heiko Ludwig, Nathalie Baracaldo, Gegi Thomas, Yi Zhou, Ali Anwar, Shashank Rajamoni, Yuya Ong, Jayaram Radhakrishnan, Ashish Verma, and Mathieu Sinn. IBM Federated Learning: an Enterprise Framework White Paper V0. 1. *CoRR*, abs/2007.10987, 2020.

[Lin17] Yehuda Lindell. How to Simulate It - A Tutorial on the Simulation Proof Technique. In *Tutorials on the Foundations of Cryptography*. Springer International Publishing, 2017.

[Lin20] Yehuda Lindell. Secure Multiparty Computation. *Communications of the ACM*, 2020.

[LPLW24] Jiacheng Liang, Ren Pang, Changjiang Li, and Ting Wang. Model Extraction Attacks Revisited. In *ASIA CCS*, 2024.

[LSPJ20] Tao Lin, Sebastian U. Stich, Kumar Kshitij Patel, and Martin Jaggi. Don't Use Large Mini-batches, Use Local SGD. In *ICLR*, 2020.

[LXC+19] Liping Li, Wei Xu, Tianyi Chen, Georgios B Giannakis, and Qing Ling. RSA: Byzantine-Robust Stochastic Aggregation Methods for Distributed Learning from Heterogeneous Datasets. In *AAAI*, 2019.

[MG06] Sergio Marti and Hector Garcia-Molina. Taxonomy of Trust: Categorizing P2P Reputation Systems. *Computer Networks*, 2006.

[MG19] Kalikinkar Mandal and Guang Gong. PrivFL: Practical Privacy-preserving Federated Regressions on High-dimensional Data over Mobile Networks. In *CCSW@CCS*, 2019.

[MMR+17] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *AISTATS*, 2017.

[MOJC23] Mohamad Mansouri, Melek Önen, Wafa Ben Jaballah, and Mauro Conti. SoK: Secure Aggregation Based on Cryptographic Schemes for Federated Learning. *PETS*, 2023.

[MR18] Payman Mohassel and Peter Rindal. ABY^3: A Mixed Protocol Framework for Machine Learning. In *ACM CCS*, 2018.

[MSS+25] Felix Marx, Thomas Schneider, Ajith Suresh, Tobias Wehrle, Christian Weinert, and Hossein Yalame. WW-FL: Secure and Private Large-Scale Federated Learning (Full Version). *CoRR*, abs/2302.09904, 2025. https://doi.org/10.48550/arXiv.2302.09904.

[MTPBH21] Christian Mouchet, Juan Troncoso-Pastoriza, Jean-Philippe Bossuat, and Jean-Pierre Hubaux. Multiparty Homomorphic Encryption from Ring-Learning-With-Errors. *PETS*, 2021.

[MWCB23] Zoltán Ádám Mann, Christian Weinert, Daphnee Chabal, and Joppe W. Bos. Towards Practical Secure Neural Network Inference: The Journey So Far and the Road Ahead. *ACM Computing Surveys*, 2023.

[MYP24] Peihua Mai, Ran Yan, and Yan Pang. RFLPA: A Robust Federated Learning Framework against Poisoning Attacks with Secure Aggregation. In *NeurIPS*, 2024.

[MZ17] Payman Mohassel and Yupeng Zhang. SecureML: A System for Scalable Privacy-Preserving Machine Learning. In *IEEE S&P*, 2017.

[NC23] Lucien K. L. Ng and Sherman S. M. Chow. SoK: Cryptographic Neural-Network Computation. In *IEEE S&P*, 2023.

[NRC+22] Thien Duc Nguyen, Phillip Rieger, Huili Chen, Hossein Yalame, Helen Möllering, Hossein Fereidooni, Samuel Marchal, Markus Miettinen, Azalia Mirhoseini, Shaza Zeitouni, Farinaz Koushanfar, Ahmad-Reza Sadeghi, and Thomas Schneider. FLAME: Taming Backdoors in Federated Learning. In *USENIX Security*, 2022.

[OA22] Ahmed El Ouadrhiri and Ahmed Abdelhadi. Differential Privacy for Deep and Federated Learning: A Survey. *IEEE Access*, 2022.

[OEC22] OECD. *Building Trust to Reinforce Democracy.* OECD, 2022.

[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*, 2018.

[Pai99] Pascal Paillier. Public-Key Cryptosystems based on Composite Degree Residuosity Classes. In *EUROCRYPT*, 1999.

[PFA22] Dario Pasquini, Danilo Francati, and Giuseppe Ateniese. Eluding Secure Aggregation in Federated Learning via Model Inconsistency. In *CCS*, 2022.

[PS20] Arpita Patra and Ajith Suresh. BLAZE: Blazing Fast Privacy-Preserving Machine Learning. In *NDSS*, 2020.

[PSSY21] Arpita Patra, Thomas Schneider, Ajith Suresh, and Hossein Yalame. ABY2.0: Improved Mixed-Protocol Secure Two-Party Computation. In *USENIX Security*, 2021.

[RAD78] Ronald L. Rivest, Len Adleman, and Michael L. Dertouzos. On Data Banks and Privacy Homomorphisms. *Foundations of Secure Computation*, 1978.

[RCF+21] Mark Russinovich, Manuel Costa, Cédric Fournet, David Chisnall, Antoine Delignat-Lavaud, Sylvan Clebsch, Kapil Vaswani, and Vikas Bhatia. Toward Confidential Cloud Computing. *Communications of the ACM*, 2021.

[RHJ+24] Ashish Rauniyar, Desta Haileselassie Hagos, Debesh Jha, Jan Erik Håkegård, Ulas Bagci, Danda B. Rawat, and Vladimir Vlassov. Federated Learning for Medical Applications: A Taxonomy, Current Trends, Challenges, and Future Research Directions. *IEEE Internet Things J.*, 2024.

[RHL+20] Nicola Rieke, Jonny Hancox, Wenqi Li, Fausto Milletarì, Holger R. Roth, Shadi Albarqouni, Spyridon Bakas, Mathieu N. Galtier, Bennett A. Landman, Klaus H. Maier-Hein, Sébastien Ourselin, Micah J. Sheller, Ronald M. Summers, Andrew Trask, Daguang Xu, Maximilian Baust, and M. Jorge Cardoso. The future of digital health with federated learning. *npj Digit. Medicine*, 3, 2020.

[RRK+20] Deevashwer Rathee, Mayank Rathee, Nishant Kumar, Nishanth Chandran, Divya Gupta, Aseem Rastogi, and Rahul Sharma. CrypTFlow2: Practical 2-Party Secure Inference. In *ACM CCS*, 2020.

[RSWP23] Mayank Rathee, Conghao Shen, Sameer Wagh, and Raluca Ada Popa. ELSA: Secure Aggregation for Federated Learning with Malicious Actors. In *IEEE S&P*, 2023.

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

[RWT+18] M. Sadegh Riazi, Christian Weinert, Oleksandr Tkachenko, Ebrahim M. Songhori, Thomas Schneider, and Farinaz Koushanfar. Chameleon: A Hybrid Secure Computation Framework for Machine Learning Applications. In *ASIA CCS*, 2018.

[SAG+23] Jinhyun So, Ramy E. Ali, Basak Güler, Jiantao Jiao, and Amir Salman Avestimehr. Securing Secure Aggregation: Mitigating Multi-Round Privacy Leakage in Federated Learning. In *AAAI*, 2023.

[SARK02] Lakshminarayanan Subramanian, Sharad Agarwal, Jennifer Rexford, and Randy H. Katz. Characterizing the Internet Hierarchy from Multiple Vantage Points. In *INFOCOM*, 2002.

[SBBE25] Jonas Sander, Sebastian Berndt, Ida Bruhns, and Thomas Eisenbarth. Dash: Accelerating Distributed Private Convolutional Neural Network Inference with Arithmetic Garbled Circuits. *IACR CHES*, 2025.

[SH21] Virat Shejwalkar and Amir Houmansadr. Manipulating the Byzantine: Optimizing Model Poisoning Attacks and Defenses for Federated Learning. In *NDSS*, 2021.

[SHKR22] Virat Shejwalkar, Amir Houmansadr, Peter Kairouz, and Daniel Ramage. Back to the Drawing Board: A Critical Evaluation of Poisoning Attacks on Production Federated Learning. In *IEEE S&P*, 2022.

[SPT+21] Sinem Sav, Apostolos Pyrgelis, Juan Ramón Troncoso-Pastoriza, David Froelicher, Jean-Philippe Bossuat, Joao Sa Sousa, and Jean-Pierre Hubaux. POSEIDON: Privacy-Preserving Federated Neural Network Learning. In *NDSS*, 2021.

[Sta22] Statista Research Department. Share of population who trust their government worldwide 2022, by country. https://www.statista.com/statistics/1362804/trust-government-world/, 2022.

[TCLY22] Zhiyi Tian, Lei Cui, Jie Liang, and Shui Yu. A Comprehensive Survey on Poisoning Attacks and Countermeasures in Machine Learning. *ACM Computing Surveys*, 2022.

[Tra20] Xavier Tracol. "Schrems II": The return of the Privacy Shield. *Computer Law & Security Review*, 2020.

[TTGL20] Vale Tolpegin, Stacey Truex, Mehmet Emre Gursoy, and Ling Liu. Data Poisoning Attacks Against Federated Learning Systems. In *ESORICS*, 2020.

[TXLZ25] Jinling Tang, Haixia Xu, Huimei Liao, and Yinchang Zhou. SeaFlame: Communication-Efficient Secure Aggregation for Federated Learning against Malicious Entities. *IACR CHES*, 2025.

[U.S22] U.S. Department of Commerce. EU-US Data Privacy Framework. https://www.dataprivacyframework.gov, 2022.

[W3C25] W3C. Federated Learning Community Group. https://www.w3.org/community/federated-learning/, 2025.

[WA16] Martin A Weiss and Kristin Archick. U.S.-EU Data Privacy: From Safe Harbor to Privacy Shield, 2016.

[WBS+24] Di Wu, Jun Bai, Yiliao Song, Junjun Chen, Wei Zhou, Yong Xiang, and Atul Sajjanhar. FedInverse: Evaluating Privacy Leakage in Federated Learning. In *ICLR*, 2024.

[WCH+24] Zhibo Wang, Zhiwei Chang, Jiahui Hu, Xiaoyi Pang, Jiacheng Du, Yongle Chen, and Kui Ren. Breaking Secure Aggregation: Label Leakage from Aggregated Gradients in Federated Learning. In *IEEE INFOCOM*, 2024.

[WGF+22] Yuxin Wen, Jonas Geiping, Liam Fowl, Micah Goldblum, and Tom Goldstein. Fishing for User Data in Large-Batch Federated Learning via Gradient Magnification. In *ICML*, 2022.

[WHZ+25] Lingling Wang, Mei Huang, Zhengyin Zhang, Meng Li, Jingjing Wang, and Keke Gai. RaSA: Robust and Adaptive Secure Aggregation for Edge-Assisted Hierarchical Federated Learning. *IEEE TIFS*, 2025.

[WSR+20] Hongyi Wang, Kartik Sreenivasan, Shashank Rajput, Harit Vishwakarma, Saurabh Agarwal, Jy-yong Sohn, Kangwook Lee, and Dimitris S. Papailiopoulos. Attack of the Tails: Yes, You Really Can Backdoor Federated Learning. In *NeurIPS*, 2020.

[WTB+21] Sameer Wagh, Shruti Tople, Fabrice Benhamouda, Eyal Kushilevitz, Prateek Mittal, and Tal Rabin. Falcon: Honest-Majority Maliciously Secure Framework for Private Deep Learning. *PETS*, 2021.

[WWCJ22] Jiayi Wang, Shiqiang Wang, Rong-Rong Chen, and Mingyue Ji. Demystifying Why Local Aggregation Helps: Convergence Analysis of Hierarchical SGD. In *AAAI*, 2022.

[WXL+21] Zhiyuan Wang, Hongli Xu, Jianchun Liu, He Huang, Chunming Qiao, and Yangming Zhao. Resource-Efficient Federated Learning with Hierarchical Aggregation in Edge Computing. In *INFOCOM*, 2021.

[XFG25] Yueqi Xie, Minghong Fang, and Neil Zhenqiang Gong. PoisonedFL: Model Poisoning Attacks to Federated Learning via Multi-Round Consistency. In *CVPR*, 2025.

[XHCL20] Chulin Xie, Keli Huang, Pin-Yu Chen, and Bo Li. DBA: Distributed Backdoor Attacks against Federated Learning. In *ICLR*, 2020.

[XHX+23] Guowen Xu, Xingshuo Han, Shengmin Xu, Tianwei Zhang, Hongwei Li, Xinyi Huang, and Robert H. Deng. Hercules: Boosting the Performance of Privacy-Preserving Federated Learning. *IEEE TDSC*, 2023.

[XXE12] Han Xiao, Huang Xiao, and Claudia Eckert. Adversarial Label Flips Attack on Support Vector Machines. In *ECAI*, 2012.

[Yan21] He Yang. H-FL: A Hierarchical Communication Efficient and Privacy-Protected Architecture for Federated Learning. In *IJCAI*, 2021.

[Yao86] Andrew Chi-Chih Yao. How to Generate and Exchange Secrets (Extended Abstract). In *FOCS*, 1986.

[YCRB18] Dong Yin, Yudong Chen, Kannan Ramchandran, and Peter L. Bartlett. Byzantine-Robust Distributed Learning: Towards Optimal Statistical Rates. In *ICML*, 2018.

[YZH22] Xuefei Yin, Yanming Zhu, and Jiankun Hu. A Comprehensive Survey of Privacy-preserving Federated Learning: A Taxonomy, Review, and Future Directions. *ACM Computing Surveys*, 2022.

[ZBA+25] Joshua C. Zhao, Saurabh Bagchi, Salman Avestimehr, Kevin S. Chan, Somali Chaterji, Dimitris Dimitriadis, Jiacheng Li, Ninghui Li, Arash Nourian, and Holger R. Roth. The Federation Strikes Back: A Survey of Federated Learning Privacy Attacks, Defenses, Applications, and Policy Landscape. *ACM Computing Surveys*, 2025.

[ZCJG22] Zaixi Zhang, Xiaoyu Cao, Jinyuan Jia, and Neil Zhenqiang Gong. FLDetector: Defending Federated Learning Against Model Poisoning Attacks via Detecting Malicious Clients. In *KDD*, 2022.

[ZIC+21] Michael Zink, David E. Irwin, Emmanuel Cecchet, Hakan Saplakoglu, Orran Krieger, Martin C. Herbordt, Michael Daitzman, Peter Desnoyers, Miriam Leeser, and Suranga Handagala. The Open Cloud Testbed (OCT): A Platform for Research into new Cloud Technologies. In *IEEE CloudNet*, 2021.

[ZLD+25] Kaixiang Zhao, Lincan Li, Kaize Ding, Neil Zhenqiang Gong, Yue Zhao, and Yushun Dong. A Survey of Model Extraction Attacks and Defenses in Distributed Computing Environments. In *KDD*, 2025.

[ZPS+22] Zhengming Zhang, Ashwinee Panda, Linyue Song, Yaoqing Yang, Michael W. Mahoney, Prateek Mittal, Kannan Ramchandran, and Joseph Gonzalez. Neurotoxin: Durable Backdoors in Federated Learning. In *ICML*, 2022.

[ZSE+24] Joshua C. Zhao, Atul Sharma, Ahmed Roushdy Elkordy, Yahya H. Ezzeldin, Salman Avestimehr, and Saurabh Bagchi. Loki: Large-scale Data Reconstruction Attack against Federated Learning through Model Manipulation. In *IEEE S&P*, 2024.

[ZWP+23] Banghua Zhu, Lun Wang, Qi Pang, Shuai Wang, Jiantao Jiao, Dawn Song, and Michael I. Jordan. Byzantine-Robust Federated Learning with Optimal Statistical Rates. In *AISTATS*, 2023.