FLSeg: Enhancing Privacy and Robustness in Federated Learning under Heterogeneous Data via Model Segmentation

Supplementary Material

A. FLSeg process

The overall workflow of FLSeg is illustrated in Algorithm 1. For simplicity, some parameters (e.g., learning rate η) are omitted to focus on the core process. In FLSeg training, the mode of segmentation is fixed and publicly known to ensure consistency between `Exchange` and `Aggregation`. The personalized model v_i^{t+1} obtained from `LocalTraining` is used solely for local tasks and does not participate in aggregation, whereas updates to the global model w_i^{t+1} are involved in subsequent `Exchange` and `Aggregation`.

Algorithm 1: FLSeg Process

```
  Input: Initial model $w^0$, training rounds $T$, the
         mode of segmentations $\{\gamma^{(1)}, \dots, \gamma^{(d)}\}$
  Output: Model after $T$ training round $w^T$
1 $\mathcal{S}$ initializes $(pk_\mathcal{S}, sk_\mathcal{S})$ and broadcasts $pk_\mathcal{S}$ to $c_i$;
2 for $t \leftarrow 0$ to $T-1$ do
3     $\mathcal{S}$ broadcasts $w^t$ to all client;
4     for each client $c_i$ parallel do
5         $v_i^{t+1}, w_i^{t+1} \leftarrow$ LocalTraining($w^t$);
6         perform Exchange($w_i^{t+1}$, $c_j$) and upload
            mixed model to $\mathcal{S}$;
7     $\mathcal{S}$ computes $w^{t+1} \leftarrow$
        Aggregation($\{(w_i^{t+1})_{mix}\}_{i\in[n]}$,
        $\{\gamma^{(1)}, \dots, \gamma^{(d)}\}$);
8 return $w^T$
```

B. Reputation System

To test the performance of the reputation system in defending against Byzantine attacks, we conducted experiments on CIFAR-10 using Min-Max attacks, with 40% of the participants being attackers. The results are shown in Figure B.1.

We recorded the similarity scores for all uploaded models calculated by the server during the first round of aggregation, with the results shown in Figure B.1 (Left). It can be observed that the mixed models resulting from Segment Exchanges between honest parties have significantly higher similarity scores compared to those exchanged with attackers. This is because the segments of malicious clients will deviate from the benchmark, resulting in lower similarity scores.

Honest clients that exchange segments with malicious users will receive low similarity scores, which will reduce the reputation of the malicious clients. To verify this, We recorded the average reputation of each client at the end of training, with the results shown in Figure B.1 (Right). The results show that the average reputation of honest clients is much higher than that of malicious clients in all clients' reputation lists. This means that malicious clients will find it increasingly difficult to match with honest clients during the Segment Exchange phase.

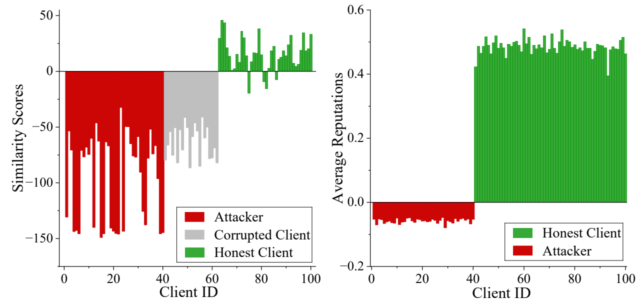

Figure B.1. Left: Similarity scores calculated by the server during the first round of aggregation. Right: Average reputation of each user at the end of training.

C. Computational Time Complexity and Overhead

In Section 5.2, we compared the computational efficiency of FLSeg with privacy-preserving FL methods in terms of privacy protection. Here, we provide the complexity analysis and experimental setup for the comparison methods.

C.1. Complexity

SecAgg [4]. The client-side computational complexity is $\mathcal{O}(n^2 + \ell n)$, including $\mathcal{O}(n)$ for key agreement with all clients, $\mathcal{O}(\ell n)$ for pairwise mask generation for the model with all other clients, and $\mathcal{O}(n^2)$ for generating t-out-of-n Shamir secret shares and communicating with all clients. On the server side, the computational complexity is $\mathcal{O}(n^2\ell)$, comprising $\mathcal{O}(n^2)$ for reconstructing the mask seeds of dropped users from the secret shares submitted by surviving users using Lagrange basis polynomials, and $\mathcal{O}(n^2\ell)$ for removing masks from each user's masked input.

Bell et al. [2] and ACORN [1]. These methods share the same privacy protection mechanism, with ACORN adding zero-knowledge proofs (ZKP) to restrict gradient norms for robustness. Since ZKP primarily ensures robustness

Dataset	**FEMNIST**	**CIFAR-10**	**CIFAR-100**
Architecture	MLP	CNN	SqueezeNet
# Parameters	84762	62006	781156
# Communication rounds	100	100	400
# Local epochs	5	5	1
Optimizer	SGD	SGD	SGD
Batch size	64	64	64
Learning rate	0.1	0.1	0.1
Momentum	0.5	0.5	0.5
Weight decay	0.0001	0.0001	0.0001
Learning rate decay	No	No	No
Gradient clipping	Yes	Yes	Yes
Clipping norm	2	2	2

Table C.1. Default experimental settings for FEMNIST, CIFAR-10 and CIFAR-100

rather than privacy, we exclude its overhead from ACORN's cost analysis. The client-side computational complexity is $\mathcal{O}(\log^2 n + \ell \log n)$, consisting of $\mathcal{O}(\log n)$ for key agreement with neighbors, $\mathcal{O}(\ell \log n)$ for pairwise mask generation, and $\mathcal{O}(\log^2 n)$ for t-out-of-n Shamir secret sharing. The server-side computational complexity is $\mathcal{O}(n(\log^2 n + \ell \log n))$, including $\mathcal{O}(n \log^2 n)$ for reconstructing pairwise masks for dropped clients, $\mathcal{O}(n\ell \log n)$ for reconstructing model masks for non-dropped clients, and $\mathcal{O}(n\ell)$ for collecting, verifying, and summing all masks.

ShieldFL [14]. These methods uses Two-Trapdoor Homomorphic Encryption based on Paillier Homomorphic Encryption. We exclude the step of computing the aggregation results on the server and focus solely on the encryption and decryption computational overhead. On the client side, the computational complexity is $\mathcal{O}(\ell(T_{\text{Enc}} + T_{\text{Dec}}))$, as clients encrypt and decrypt their models using Paillier Homomorphic Encryption. On the server side, the computational complexity is $\mathcal{O}(\ell(T_{\text{Dec}}))$, as the server performs partial decryption on the uploaded models.

PEFL [11]. On the client side, the computational complexity is $\mathcal{O}(\ell(T_{\text{Enc}} + T_{\text{Dec}}))$, as clients use Paillier Homomorphic Encryption to encrypt and decrypt their models. On the server side, the complexity is $\mathcal{O}(\ell n(T_{\text{Enc}} + T_{\text{Dec}}))$, as the dual-server system repeatedly encrypts and decrypts models during the computation of Pearson coefficients and secure mean values.

C.2. Experimental Setup

We tested each method's runtime in a practical setting with $n = 10^3$ and $\ell = 10^6$. For SecAgg, Bell et al., ACORN, and the Diffie-Hellman key agreement in FLSeg, we use the elliptic curve curve25519 with the modulus q set as the group size. The t-out-of-n secret sharing scheme is configured with $t = 0.1n$, and we assume that 10% of the users drop out. For ShieldFL and PEFL, which both utilize Paillier Homomorphic Encryption, the key length is set to 128 bits.

C.3. Communication Overhead

The client communication cost in FLSeg is $O(2\ell)$, accounting for Segment Exchange and model upload. HE-based methods [11, 14] introduce ∼8× overhead due to ciphertext size $O(\ell \cdot |X|)$ with $|X| \geq 256$ bits. SMPC-based methods [1, 4, 13] have similar complexity $O(2\ell + 5 \log n)$ but require five rounds with neighbors for key agreement, making them slightly more costly in practice. FLSeg is more communication-efficient.

D. Setups for Experiments

D.1. Datasets and Models

The setups for datasets FEMNIST [5], CIFAR-10 [9] and CIFAR-100 [9] are listed in Tab. C.1.

D.2. Evaluated attacks

We simulated three types of Byzantine attacks: Labelflipping [7], Min-Max [17], IPM [18], and two privacy inference attacks: Inverting-Gradients [8] and iDGL [19]. The parameters for all attacks are shown in Table D.2.

To maintain consistency with the notation used in the original papers of the compared methods, we slightly abuse some symbols here. The same symbol may represent different meanings in different methods, and the specific meanings can be referenced in the original papers.

For Labelflipping, we flipped the labels of all attackers' data, i.e., if the labels in the dataset are within the range $[0, m]$, then for each class c, the label is set to $m - c$.

Attacks	Hyperparameters
Labelflipping	N/A
Min-Max	$\gamma_{\text{init}} = 5, \tau = 1 \times 10^{-2}$, set δ as the coordinate-wise standard deviation
IPM	$\epsilon = 10$
Inverting-Gradients	learning rate 0.1
iDLG	learning rate 1

Table D.2. Attacks and Hyperparameters. N/A indicates that the attack has no hyperparameters that need to be set.

D.3. Compared Methods

We compared our method with several existing baselines, including the FL baseline FedAvg [15]; Byzantine-robust FL methods: Krum [3], DnC [17], RFA [16], GAS [12], FLTrust [6], and ClippedClustering [10]; and privacy-preserving robust FL methods: ACORN [1], ShieldFL [14], and PEFL [11]. The hyperparameters for all Compared methods are shown in Table D.3. Consistent with Sec. D.2, we slightly abuse some symbols here.

Compared Methods	Hyperparameters
FedAvg	N/A
Krum	N/A
DnC	niters $= 1, c = 2.0, b = 200$
RFA	$R = 100, \nu = 1e6$
GAS	$p = 200$
FLTrust	N/A
ClippedClustering	τ is the median of L_2 Norm clustering with average linkage
ACORN	L_2 Norm Bound $B = 4$ for CIFAR-100 and FEMNIST, $B = 3$ for CIFAR-10
ShieldFL	N/A
PEFL	N/A
FLSeg(Krum)	segment size $\|\gamma\| = 200, \lambda = 0.1$
FLSeg(Dnc)	$\|\gamma\| = 200, \lambda = 0.1$, niters $= 1, c = 2.0, b = \|\gamma\|$

Table D.3. Compared methods and Hyperparameters. N/A indicates that the attack has no hyperparameters that need to be set.

In our experiments, FLSeg divides the model into segments of 200 consecutive parameters per group, with the remaining parameters forming the final group.

When applying RAR, methods such as Krum, RFA, and Median, which originally operate on the entire model, are directly applied to each segment. For DnC, which requires subsampling, the sampling size is set to be the same as the segment size.

D.4. Segment size Experiment Setup

In the impact of segment size experiments, the original CNN trained on CIFAR-10 had only 62,006 parameters. To more precisely investigate the impact of segment size on privacy protection, we use a larger CNN with $\ell = 2,904,970$ parameters. Except for model parameters, all other experimental settings remain the same as in Tab. C.1.

D.5. Computing Infrastructure

The experiments were carried out on a computing environment equipped with an Intel(R) Xeon(R) Gold 6133 CPU running at 2.50GHz, paired with an NVIDIA GeForce RTX 3090 GPU. The system was supported by 128GB of DDR4 RAM and utilized a 1TB SSD for storage. The operating system used was Ubuntu 20.04 LTS. The experiments were conducted using Python 3.10 and PyTorch 2.0.1, with CUDA 11.7 providing GPU acceleration. This setup was consistently used throughout the experiments to ensure reliable and reproducible results.

References

[1] James Bell, Adrià Gascón, Tancrède Lepoint, Baiyu Li, Sarah Meiklejohn, Mariana Raykova, and Cathie Yun. ACORN: input validation for secure aggregation. In *32nd USENIX Security Symposium, USENIX Security 2023, Anaheim, CA, USA, August 9-11, 2023*, pages 4805–4822. USENIX Association, 2023. 1, 2, 3

[2] James Henry Bell, Kallista A. Bonawitz, Adrià Gascón, Tancrède Lepoint, and Mariana Raykova. Secure single-server aggregation with (poly)logarithmic overhead. In *CCS '20: 2020 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, USA, November 9-13, 2020*, pages 1253–1269. ACM, 2020. 1

[3] Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. Machine learning with adversaries: Byzantine tolerant gradient descent. *Advances in neural information processing systems*, 30, 2017. 3

[4] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1175–1191, 2017. 1, 2

[5] Sebastian Caldas, Sai Meher Karthik Duddu, Peter Wu, Tian Li, Jakub Konečný, H Brendan McMahan, Virginia Smith, and Ameet Talwalkar. Leaf: A benchmark for federated settings. *arXiv preprint arXiv:1812.01097*, 2018. 2

[6] Xiaoyu Cao, Minghong Fang, Jia Liu, and Neil Zhenqiang Gong. Fltrust: Byzantine-robust federated learning via trust bootstrapping. In *NDSS*. The Internet Society, 2021. 3

[7] Minghong Fang, Xiaoyu Cao, Jinyuan Jia, and Neil Gong. Local model poisoning attacks to {Byzantine-Robust} federated learning. In *29th USENIX security symposium (USENIX Security 20)*, pages 1605–1622, 2020. 2

[8] Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. Inverting gradients-how easy is it to break

privacy in federated learning? *Advances in neural information processing systems*, 33:16937–16947, 2020. 2

[9] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 2

[10] Shenghui Li, Edith C-H Ngai, and Thiemo Voigt. An experimental study of byzantine-robust aggregation schemes in federated learning. *IEEE Transactions on Big Data*, 2023. 3

[11] Xiaoyuan Liu, Hongwei Li, Guowen Xu, Zongqi Chen, Xiaoming Huang, and Rongxing Lu. Privacy-enhanced federated learning against poisoning adversaries. *IEEE Transactions on Information Forensics and Security*, 16:4574–4588, 2021. 2, 3

[12] Yuchen Liu, Chen Chen, Lingjuan Lyu, Fangzhao Wu, Sai Wu, and Gang Chen. Byzantine-robust learning on heterogeneous data via gradient splitting. In *International Conference on Machine Learning*, pages 21404–21425. PMLR, 2023. 3

[13] Hidde Lycklama, Lukas Burkhalter, Alexander Viand, Nicolas Küchler, and Anwar Hithnawi. Rofl: Robustness of secure federated learning. In *SP*, pages 453–476. IEEE, 2023. 2

[14] Zhuoran Ma, Jianfeng Ma, Yinbin Miao, Yingjiu Li, and Robert H Deng. Shieldfl: Mitigating model poisoning attacks in privacy-preserving federated learning. *IEEE Transactions on Information Forensics and Security*, 17:1639–1654, 2022. 2, 3

[15] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017. 3

[16] Krishna Pillutla, Sham M Kakade, and Zaid Harchaoui. Robust aggregation for federated learning. *IEEE Transactions on Signal Processing*, 70:1142–1154, 2022. 3

[17] Virat Shejwalkar and Amir Houmansadr. Manipulating the byzantine: Optimizing model poisoning attacks and defenses for federated learning. In *NDSS*, 2021. 2, 3

[18] Cong Xie, Oluwasanmi Koyejo, and Indranil Gupta. Fall of empires: Breaking byzantine-tolerant sgd by inner product manipulation. In *Uncertainty in Artificial Intelligence*, pages 261–270. PMLR, 2020. 2

[19] Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. idlg: Improved deep leakage from gradients. *CoRR*, abs/2001.02610, 2020. 2