PrivateBoost: Privacy-Preserving Federated Gradient Boosting for Cross-Device Medical Data

Bernhard Specht*[1,2], Samaher Garbaya[1], Orhan Ermis[3], Reinhard Schneider[4], Ricardo Chavarriaga[5], Djamel Khadraoui[3], and Zied Tayeb[1,6]

[1]MyelinZ, Manchester, UK / San Francisco, USA / Leuven, Belgium /Luxembourg
[2]University of Luxembourg, Esch-sur-Alzette, Luxembourg
[3]Luxembourg Institute of Science and Technology (LIST), Esch-sur-Alzette, Luxembourg
[4]Luxembourg Centre for Systems Biomedicine (LCSB), Esch-sur-Alzette, Luxembourg
[5]ZHAW Zurich University of Applied Sciences, Winterthur, Switzerland
[6]University of Lincoln, Lincoln, United Kingdom

Abstract

Cross-device medical federated learning where individual patients participate directly rather than institutions poses a unique challenge: each client holds only a few samples, often just one (e.g., a single diagnostic record), leaving insufficient local data for gradient computation. Existing approaches, such as Secure Aggregation, require client-to-client coordination impractical for intermittently available mobile devices, while homomorphic encryption-based alternatives introduce sophisticated key management and coordination requirements ill-suited to dynamic cross-device deployments. We present *privateboost*, a federated XGBoost system that addresses this setting through m-of-n Shamir secret sharing with commitment-based anonymous aggregation. Clients distribute shares to a fixed set of shareholders requiring no client-to-client communication and the aggregator reconstructs only aggregate gradient sums via Lagrange interpolation, never observing individual values or client identities. We evaluate on UCI medical datasets, demonstrating 98% split gain retention relative to centralized XGBoost and accuracy resilient to up to 80% client dropout.

1 Introduction

Federated learning enables machine learning on distributed data without centralizing raw samples [1]. However, much existing work focuses on a *cross-silo* setting where each client (typically an organization) holds multiple data samples and can perform local training [2]. Popular frameworks such as Flower [3] and NVIDIA FLARE [4] support both settings but are primarily evaluated in cross-silo or simulated deployments.

In medical settings, "federated learning" typically means hospitals or research institutions each manage patient cohorts [5, 6], with the institution serving as a trusted aggregation point for its patients' data. True *cross-device* medical federated learning, where individual patients control their own records without institutional intermediaries, remains largely unexplored. Cross-device settings have been studied for applications like mobile keyboard prediction [7], but medical applications present unique challenges.

First, *extreme non-IID data*: each client holds only a few samples, often just one. A patient participating in a federated study may contribute only their single health record. There are no local batches large enough for meaningful gradient computation; each participant is effectively a single data point. Second, tree-based methods like XGBoost [8] require global statistics (gradient histograms across all samples) to find optimal split points. Without local batches large enough for meaningful statistics or secure aggregation, clients cannot contribute to these histograms without revealing their individual values.

Existing privacy-preserving approaches face practical barriers in this setting. SecAgg [9] requires client-to-client coordination for pairwise key agreement, which is problematic when clients are intermittently online mobile devices. Additionally, ho-

momorphic encryption-based solutions can serve as alternative secure aggregation mechanisms, namely multi-key and multi-party homomorphic encryption schemes [10, 11]. However, such mechanisms require sophisticated key management procedures, including distributed key generation, key switching, and evaluation key distribution. Therefore, homomorphic encryption-based schemes introduce additional protocol complexity and coordination requirements, which may be difficult to satisfy in highly dynamic cross-device deployments with frequently joining and leaving clients.

We present *privateboost*, a privacy-preserving federated XGBoost system designed for the one-sample-per-client setting:

1. A protocol using m-of-n Shamir secret sharing with commitment-based client anonymity
2. A three-party architecture (clients → shareholders → aggregator) requiring no client-to-client communication
3. Privacy guarantees: raw data never leaves the client; shareholders see only shares; the aggregator sees only aggregate sums. Hence, under the non-collusion assumption, where fewer than m shareholders collude, individual gradients shared by clients are information-theoretically secure
4. Empirical validation on medical datasets demonstrating 98% split gain retention with dropout resilience

2 Related Work

2.1 Secure Aggregation

SecAgg [9] enables a server to compute sums of client updates without learning individual values. The protocol uses pairwise masking: each pair of clients agrees on a random mask via Diffie-Hellman key exchange, and these masks cancel when the server sums all contributions. This requires $O(n)$ pairwise key agreements per client and coordination to handle dropouts. Subsequent work has reduced communication to polylogarithmic overhead [12], but the protocols still rely on structured communication patterns between clients.

Our approach uses a star topology where clients distribute shares to a fixed set of shareholders without any client-to-client communication. This is better suited to cross-device settings where clients are intermittently available.

2.2 Federated Gradient Boosting

Gradient boosting [13] builds ensembles of weak learners. LightGBM [14] uses histogram-based split finding for efficiency; XGBoost [8] supports both exact and approximate histogram methods.

SecureBoost [15] addresses federated XGBoost in the *vertical* federated learning setting, where different parties hold different features for the same samples. This assumes institutional participants with aligned datasets. Federated Forest [16] similarly targets cross-silo settings with multiple samples per participant. Recent work has extended secret sharing to vertical federated XGBoost [17], and FedTree [18] provides a comprehensive system supporting both horizontal and vertical settings with configurable privacy techniques.

For *horizontal* federated learning, SimFL [19] uses locality-sensitive hashing (LSH) to share similarity information between parties. However, SimFL's design assumes each party holds many samples: each party builds trees using only its local instances, weighted by gradients from similar samples at other parties. This fundamentally precludes the one-sample-per-client setting. Additionally, LSH-based approaches have weaker privacy guarantees than cryptographic methods: parties learn which samples are similar across institutions, which could enable inference attacks on sensitive medical data.

Differential privacy has also been applied to gradient boosting [20] in centralized settings, but these approaches assume access to the full dataset and do not address the federated or cross-device setting.

Table 1 summarizes the key properties of existing federated tree-learning approaches. Our work addresses the orthogonal *horizontal* federated setting with extreme non-IID: one sample per client, providing information-theoretic security via secret sharing rather than the computational security of LSH-based approaches.

2.3 Secret Sharing in Machine Learning

Shamir secret sharing [21] is a foundational primitive for multi-party computation (MPC) [22]. Verifiable extensions [23] enable detection of malicious behavior. These techniques have been applied in MPC frameworks for machine learning inference, but not extensively explored for federated training.

The key property we exploit is *additive homomorphism*: the sum of shares equals shares of the sum, enabling aggregation without reconstruction of individual values.

Table 1: Comparison of federated tree-learning approaches. HE = homomorphic encryption, SA = secure aggregation, DP = differential privacy, LSH = locality-sensitive hashing, SS = secret sharing, C2C = client-to-client, X-device = cross-device.

Method	**Setting**	**Privacy**	**Security**	**Min. Samples**	**No C2C**	**X-device**
SecureBoost [15]	Vertical	HE (Paillier)	Computational	Many	✓	–
Fed. Forest [16]	Vertical	Encoding + encryption	Computational	Many	✓	–
SimFL [19]	Horizontal	LSH	Computational	Many	–	–
Xie et al. [17]	Vertical	Secret sharing	Info-theoretic	Many	–	–
FedTree [18]	Both	HE / SA / DP	Configurable	Many	✓	–
privateboost (ours)	Horizontal	Shamir SS	Info-theoretic	**1**	✓	✓

3 Protocol

3.1 Background: Histogram-Based Split Finding

XGBoost [8] builds decision trees by greedily selecting splits that maximize a gain function. For each node, the algorithm considers every feature and every possible split threshold. To make this tractable, histogram-based methods discretize continuous features into B bins and compute gradient statistics per bin.

Each sample i contributes a gradient $g_i = \partial L/\partial \hat{y}_i$ and Hessian $h_i = \partial^2 L/\partial \hat{y}_i^2$ based on the current prediction $\hat{y}_i$ and true label y_i. For a candidate split that partitions samples into left (L) and right (R) sets, the gain is:

$$\text{Gain} = \frac{1}{2}\left[\frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{(G_L+G_R)^2}{H_L+H_R+\lambda}\right] \tag{1}$$

where $G_L = \sum_{i\in L} g_i$, $H_L = \sum_{i\in L} h_i$, and λ is a regularization parameter. The key insight is that split finding requires only *aggregate* gradient sums per bin, not individual gradients. Our protocol computes these sums without revealing individual contributions.

3.2 System Model

We consider c clients, each holding a single labeled sample (x_i, y_i). A set of n *shareholders* (e.g., $n = 3$) act as intermediate aggregation points. A single *aggregator* coordinates the training process. Figure 1 illustrates the overall system architecture.

The adversarial assumptions underlying this architecture are detailed in the threat model (Section 4.1).

3.3 Building Blocks

Shamir Secret Sharing. To share a secret value s, sample a random polynomial $f(x)$ of degree $m-1$ with $f(0) = s$. Distribute shares $(i, f(i))$ for $i = 1, \ldots, n$ to n shareholders. Any m shares can reconstruct s via Lagrange interpolation; fewer than m shares reveal nothing about s.

Additive Homomorphism. For shares of values a and b, the sum of shares equals shares of $(a + b)$. This allows shareholders to sum their received shares without learning individual values, and the aggregator reconstructs only the sum.

Commitments. Each client generates a commitment $c = H(\text{round}\|\text{client_id}\|\text{nonce})$ using a fresh nonce per round. Shareholders only sum shares that have matching commitment hashes, ensuring consistent aggregation: if a client's share reaches some shareholders but not others, those shares are excluded rather than causing corrupted reconstruction. The aggregator sees only commitment hashes, not client identifiers.

3.4 Protocol Flow

The protocol has two phases: a one-time statistics phase to define histogram bins, followed by repeated gradient phases to build trees.

Phase 1: Statistics. Before training, the aggregator needs feature statistics to define histogram bin edges. Each client i creates Shamir shares of their feature values $x_{i,f}$ and squared values $x_{i,f}^2$ for each feature f, along with a commitment. Clients send shares to shareholders. The aggregator requests sums from shareholders, reconstructs $\sum x$ and $\sum x^2$, computes per-feature mean μ and variance σ^2, and defines B equal-width bins spanning $\mu \pm 3\sigma$. The aggregator broadcasts bin edges to all clients.

Phase 2: Gradient Rounds. For each tree and each depth level, the aggregator needs gradient sums per (feature, bin) combination. Each client i:

1. Computes their gradient g_i and Hessian h_i from current ensemble prediction
2. For each feature f, determines which bin b their value $x_{i,f}$ falls into
3. Creates Shamir shares of (g_i, h_i) with a fresh commitment
4. Sends shares to shareholders, tagged with $(f, b, \text{commitment})$

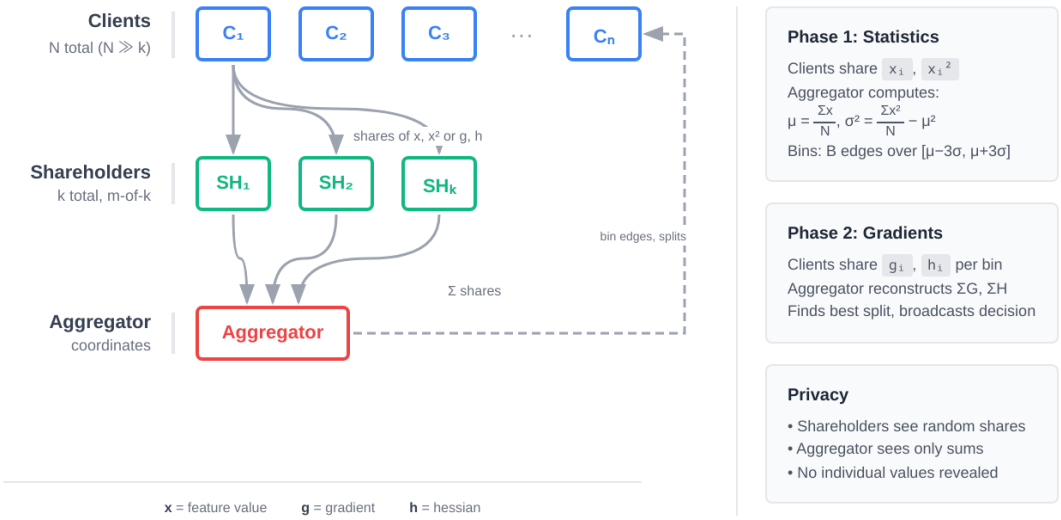

Figure 1: System architecture. Clients distribute Shamir shares to all shareholders. Shareholders sum received shares and forward partial sums to the aggregator, which reconstructs aggregate statistics (Σx, Σx^2 for binning) and gradient sums (ΣG, ΣH for splits). The aggregator broadcasts bin configurations and split decisions back to clients.

Shareholders accumulate shares grouped by (feature, bin, commitment). When the aggregator requests sums for a set of commitments, each shareholder returns summed shares per (feature, bin). The aggregator reconstructs gradient sums $G_{f,b} = \sum g_i$ and $H_{f,b} = \sum h_i$ for each bin, evaluates the gain formula for each possible split, selects the best split, and broadcasts the decision (feature, threshold) to clients. Clients update their node assignments, and the process repeats for the next depth level.

After reaching maximum depth, leaf values are computed from gradient sums and the tree is added to the ensemble. The entire gradient phase repeats for each tree.

4 Security Analysis

4.1 Threat Model

We assume an honest-but-curious (semi-honest) adversary model: all parties follow the protocol correctly but may attempt to learn additional information from their observations.

Shareholders. At most $m - 1$ out of n shareholders may collude for an m-of-n threshold scheme. The collusion threshold is configurable: increasing from 2-of-3 to 3-of-5 or higher provides stronger guarantees while maintaining dropout tolerance. Shareholders enforce a minimum-client threshold: they refuse to release aggregate sums if fewer than a configurable minimum number of clients contributed shares, preventing reconstruction from insufficiently anonymized aggregates. Crucially, shareholders do not collude with the aggregator; any such collusion would undermine the separation between share-level and aggregate-level views that the protocol relies on.

Aggregator. A single aggregator coordinates the protocol. It follows the protocol honestly but may attempt to learn individual values from the aggregates it reconstructs. It does not have access to raw shares or client identifiers.

Clients. Clients are assumed honest-but-curious. The protocol does not include cryptographic verification of individual contributions; defenses against malicious clients are discussed in Section 4.4.

4.2 Privacy Guarantees

Clients hold their raw feature values and labels locally. Each client submits Shamir shares along with a pseudonymous client identifier (to prevent duplicate submissions) to shareholders. This identifier is linked to an account but does not reveal the client's real identity. Since shares are information-theoretically secure, no single shareholder can learn anything about the original values.

Shareholders receive shares tagged with client identifiers and commitment hashes. They use identifiers to prevent duplicate submissions but do not forward them to the aggregator. Under the collusion threshold defined in the threat model, shareholders cannot reconstruct individual values. Shareholders sum

shares for requested commitments and return only these aggregated sums.

The aggregator receives summed shares from shareholders and reconstructs aggregate gradient sums via Lagrange interpolation. Crucially, it never sees client identifiers, only opaque commitment hashes. It learns split thresholds (necessary for prediction) and gradient sums per bin. It learns how many clients fall into each tree branch, but not which specific clients.

4.3 Communication and Computation Cost

Let c denote the number of clients, F the number of features, B the number of bins, n the number of shareholders, m the reconstruction threshold, T the number of trees, and d the maximum tree depth.

Communication. In the one-time statistics phase, each client creates Shamir shares of $2F$ values ($x_{i,f}$ and $x_{i,f}^2$) and distributes them to n shareholders, yielding $O(Fcn)$ total shares. In each gradient round (one depth level of one tree), each client shares $2F$ gradient values (g_i and h_i per feature), producing $O(Fcn)$ shares per round. Over T trees at depth d, total client-to-shareholder communication is $O(TdFcn)$. Each shareholder returns $O(FB)$ partial sums to the aggregator per round, giving $O(TdFBn)$ shareholder-to-aggregator communication. The aggregator broadcasts a single split decision per round.

Computation. Creating a Shamir share requires evaluating a degree-$(m-1)$ polynomial at n points, costing $O(nm)$ per value. Each client thus performs $O(Fnm)$ work per gradient round, or $O(TdFnm)$ total. Shareholders sum $O(c)$ shares per (feature, bin) bucket. The aggregator performs Lagrange interpolation over m points for each of the $O(FB)$ aggregate values per round, costing $O(TdFBm^2)$ total. Split finding adds $O(TdFB)$.

Scalability in n. Communication grows linearly in n (more shares to distribute and collect). Computation for sharing grows as $O(nm)$, and reconstruction depends only on m, not n. Increasing n while keeping m fixed improves fault tolerance (more shareholders can drop out) without affecting reconstruction cost. Increasing m strengthens the collusion threshold at the cost of higher-degree polynomials for sharing and $O(m^2)$ Lagrange reconstruction.

4.4 Limitations

Branch counts. The aggregator learns client counts per tree branch. For branches with very few clients, this narrows the anonymity set. Combined with auxiliary information, this could enable inference attacks. This motivates k-anonymous tree constraints (Section 6.2).

Shareholder path visibility. Shareholders observe which bin each client submits gradients to, revealing individual tree paths. This can be addressed using the path hiding technique described in Section 6.3: clients submit shares for all possible paths, with only the true path containing non-zero values. However, this incurs communication overhead that grows exponentially with tree depth ($O(2^d)$ per client per round).

Cross-round linkability. Commitment hashes are fresh per round (using a new random nonce each time), preventing the aggregator from linking submissions across rounds. However, shareholders can link a client's submissions across rounds via transport-layer identity (e.g., network address). This is a transport-level concern, not a protocol-level one, and is straightforwardly mitigated by routing client submissions through an anonymous communication channel.

No differential privacy. The protocol reveals exact aggregate sums. Future work addresses formal differential privacy guarantees.

Malicious clients. The honest-but-curious assumption does not protect against gradient poisoning or Sybil attacks. In practice, mobile deployments can leverage platform attestation mechanisms (such as App Check tokens on Android and App Attest on iOS) to verify that participating clients are running legitimate, unmodified application code, providing a practical defense without requiring cryptographic verification of individual contributions.

Malicious aggregator. A malicious aggregator could corrupt split decisions or model outputs. This can be mitigated by running multiple independent aggregators that cross-verify results, or by using verifiable computation techniques.

5 Experiments

We evaluate on three medical datasets from the UCI Machine Learning Repository [24]. Heart Disease [25] (303 samples; 297 used after removing incomplete records) predicts coronary artery disease from 13 clinical features. Breast Cancer [26] (569 samples) classifies tumors using 30 cell nuclei measurements. Pima Diabetes [27] (768 samples) predicts diabetes onset from 8 diagnostic measurements. We use an 80/20 train/test split for all datasets.

Following the protocol design, each client holds exactly one sample: we instantiate one client per training example. We use a 2-of-3 Shamir threshold, 10 bins per feature, max depth 3, 15 trees, learning rate 0.15, and regularization $\lambda = 2.0$. Bins are equal-width, spanning $\mu \pm 3\sigma$ computed from aggregated statistics. We compare against XGBoost with matched hyperparameters and XGBoost with default settings.

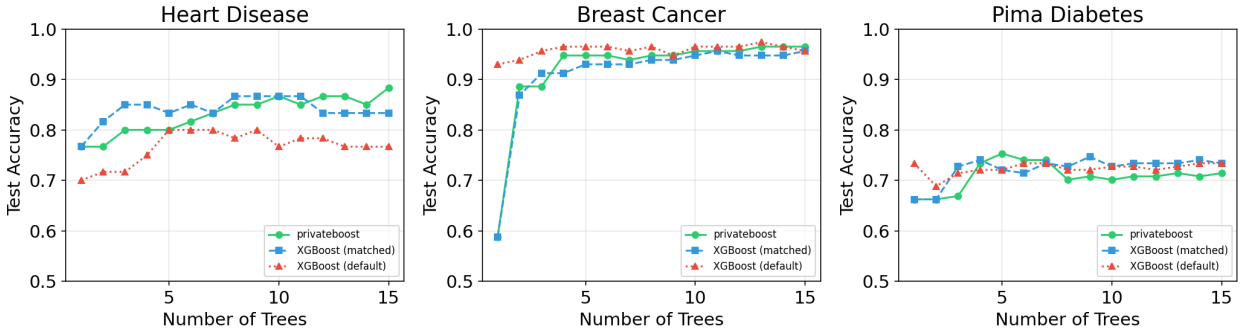

Figure 2: Test accuracy vs. number of trees for *privateboost*, XGBoost with matched hyperparameters, and XGBoost with defaults across three medical datasets.

Across all three datasets, *privateboost* achieves competitive test accuracy (Figure 2). On Heart Disease, *privateboost* achieves 88.3% compared to 83.3% for XGBoost with matched hyperparameters and 76.7% for XGBoost with defaults. On Breast Cancer, all three methods achieve 95.6% accuracy. On Pima Diabetes, *privateboost* achieves 71.4% compared to 73.4% for both XGBoost configurations.

The Heart Disease result is notable: *privateboost* outperforms both XGBoost configurations, which we attribute to a regularization effect from histogram binning that reduces overfitting on smaller datasets.

Split gain retention. To quantify the information loss from histogram binning, we define *split gain retention* per feature f as:

$$R_f = \frac{G_f^{\text{hist}}}{G_f^{\text{exact}}} \tag{2}$$

where G_f^{exact} is the optimal split gain obtained by exhaustive search over all unique thresholds in feature f (equivalent to centralized XGBoost with no binning), and G_f^{hist} is the best split gain when restricted to the B histogram bin edges derived from the privacy-preserving statistics phase. Both gains use the same gradient and Hessian values, evaluated at the root level using initial predictions. We report mean retention $\bar{R} = \frac{1}{F}\sum_{f=1}^{F} R_f$ across all F features.

On Heart Disease, mean retention is $\bar{R} = 98.1\%$ across 13 features, with 10 features achieving $R_f = 100\%$ (Figure 3). Histogram bins are constructed using aggregated statistics from the first protocol round, requiring no additional privacy cost. Features with skewed distributions may have suboptimal bin placement, with most data concentrated in a few bins. This explains the 90–93% retention observed on features like `oldpeak`.

In cross-device settings, clients are intermittently available: mobile devices go offline, patients miss check-ins, or network connectivity fails. The protocol must tolerate such dropout without requiring all clients to participate in every round.

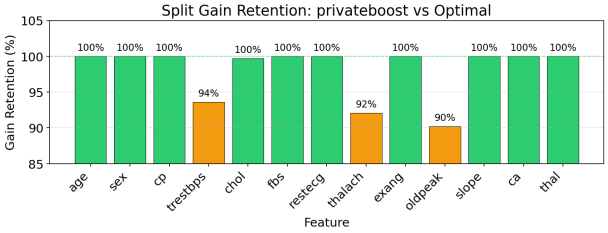

Figure 3: Split gain retention R_f per feature on Heart Disease. Mean retention $\bar{R} = 98\%$ across 13 features.

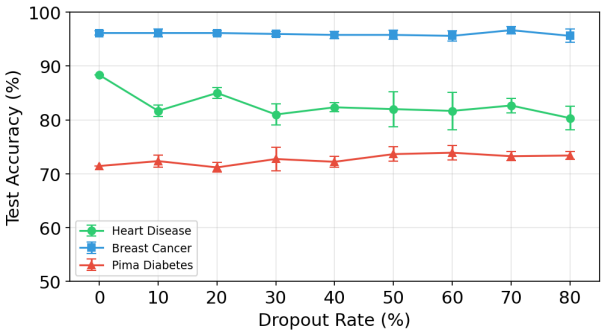

Figure 4: Test accuracy under varying client dropout rates.

The protocol naturally handles client dropout: the aggregator reconstructs from whichever clients participate in each round. We simulate dropout by randomly excluding a fraction of clients each training round, with independent sampling per round.

Impact on reconstruction. Client dropout does not affect the secret sharing reconstruction itself. Shamir reconstruction requires m out of n *shareholder* responses, which is independent of client participation. Dropout reduces the number of shares each shareholder receives, but shareholders can still

sum any subset of received shares and the aggregator can reconstruct the aggregate of participating clients. The minimum-client threshold enforced by shareholders (Section 4.1) provides a lower bound: if fewer than the minimum number of clients participate, shareholders refuse to release sums, preventing reconstruction from insufficiently anonymized aggregates.

Impact on model quality. As shown in Figure 4, accuracy remains stable up to 80% dropout across all datasets. At moderate dropout rates (20–50%), the model sees different subsets of clients per round, which can act as a form of bagging and may improve generalization. Beyond 80% dropout, accuracy degrades more sharply as gradient sums become noisy estimates computed from very few clients. The degradation is dataset-dependent: smaller datasets (Heart Disease, 297 samples) are more sensitive to extreme dropout than larger ones (Pima Diabetes, 768 samples).

6 Future Work

6.1 Differential Privacy

Adding calibrated Gaussian noise to gradient sums before reconstruction would provide formal (ε, δ)-differential privacy [28]. Similar techniques have been applied to deep learning [29]. The noise scale can be calibrated using the mean and variance statistics already computed during the binning phase, incurring no additional privacy cost. We suggest per-tree budgets of $\varepsilon = 1.0$–2.0 with composition accounting across the ensemble.

6.2 K-Anonymous Tree Structure

To prevent membership inference via branch analysis, splits could be constrained to ensure at least k clients in every resulting node, inspired by k-anonymity principles [30]. This extends the existing minimum-client enforcement in shareholders to the tree structure itself. Splits that would create nodes with fewer than k clients would be rejected, trading some model expressiveness for stronger privacy guarantees.

6.3 Private Path Hiding

In the current protocol, the aggregator learns how many clients traverse each branch of the tree, even though it cannot identify which specific clients. This branch count information could narrow the anonymity set, particularly for branches with few clients.

A stronger privacy guarantee would hide even the branch membership. At each tree depth, clients secret-share a binary indicator for each active node: 1 if the client belongs to that node, 0 otherwise. Shareholders sum these indicators alongside the gradient shares. The aggregator reconstructs only the total count per node, learning nothing about individual assignments beyond what the count reveals.

This extension incurs communication overhead of $O(2^d \times n \times c)$ shares per gradient round, where d is tree depth, n is shareholders, and c is clients. For $d = 3$, $n = 3$, and $c = 200$:

- Current protocol: $200 \times 3 = 600$ shares per round
- With path hiding: $8 \times 200 \times 3 = 4{,}800$ shares per round

This 8× overhead grows exponentially with depth, making the extension most practical for shallow trees ($d \leq 3$), which are common in gradient boosting. For applications with stringent privacy requirements, this tradeoff may be acceptable.

7 Conclusion

We presented *privateboost*, a privacy-preserving federated XGBoost system for the extreme non-IID setting where each client holds only a few samples, often just one. Our protocol ensures raw data never leaves the client, requires no client-to-client coordination, and provides resilience to participant dropout.

Evaluation on medical datasets demonstrates 98% split gain retention with accuracy comparable to centralized XGBoost. The system enables true cross-device medical federated learning where patients can participate directly without institutional intermediaries.

Acknowledgement

The authors would like to thank the Ministry of the Economy in Luxembourg and its digital health directorate for co-supporting this research. Similarly, we would like to thank the Luxembourg National Research Fund (FNR) for partially funding this research.

Funding

This work was supported in part by a PhD grant from the Luxembourg National Research Fund (FNR) under the project reference 17223919/MMS/Industrial Fellowship, and in part by the R-MMS clinical trial.

References

[1] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In

Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, volume 54 of *PMLR*, pages 1273–1282, 2017. URL `https://proceedings.mlr.press/v54/mcmahan17a.html`.

[2] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, et al. Advances and open problems in federated learning. *Foundations and Trends in Machine Learning*, 14(1–2):1–210, 2021. doi: 10.1561/2200000083.

[3] Daniel J Beutel, Taner Topal, Akhil Mathur, Xinchi Qiu, Javier Fernandez-Marques, Yan Gao, Lorenzo Sani, Kwing Hei Li, Titouan Parcollet, Pedro Porto Buarque de Gusmao, and Nicholas D Lane. Flower: A friendly federated learning research framework. arXiv preprint arXiv:2007.14390, 2020.

[4] Holger R Roth, Yan Cheng, Yuhong Wen, Ahmed Shaker, et al. NVIDIA FLARE: Federated learning from simulation to real-world. arXiv preprint arXiv:2210.13291, 2022.

[5] Nicola Rieke, Jonny Hancox, Wenqi Li, Fausto Milletari, Holger R Roth, Shadi Albarqouni, Spyridon Bakas, et al. The future of digital health with federated learning. *NPJ Digital Medicine*, 3(1):119, 2020. doi: 10.1038/s41746-020-00323-1.

[6] Micah J Sheller, Brandon Edwards, G Anthony Reina, Jason Martin, Sarthak Pati, Aikaterini Kotrotsou, Mikhail Milchenko, et al. Federated learning in medicine: facilitating multi-institutional collaborations without sharing patient data. *Scientific Reports*, 10(1):12598, 2020. doi: 10.1038/s41598-020-69250-1.

[7] Andrew Hard, Kanishka Rao, Rajiv Mathews, Swaroop Ramaswamy, Françoise Beaufays, Sean Augenstein, Hubert Eichner, Chloé Kiddon, and Daniel Ramage. Federated learning for mobile keyboard prediction. arXiv preprint arXiv:1811.03604, 2018.

[8] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794, 2016. doi: 10.1145/2939672.2939785.

[9] 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. doi: 10.1145/3133956.3133982.

[10] Jiahui Wu, Tiecheng Sun, Fucai Luo, Haiyan Wang, and Weizhe Zhang. Secure multi-key homomorphic encryption with application to privacy-preserving federated learning, 2025. URL `https://arxiv.org/abs/2506.20101`.

[11] Jung Hee Cheon, Hyeongmin Choe, Seunghong Kim, and Yongdong Yeo. Multi-party homomorphic encryption with dynamicity and ciphertext reusability, 2025. URL `https://eprint.iacr.org/2025/581`.

[12] James Henry Bell, Kallista A Bonawitz, Adrià Gascón, Tancrède Lepoint, and Mariana Raykova. Secure single-server aggregation with (poly) logarithmic overhead. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 1253–1269, 2020. doi: 10.1145/3372297.3417885.

[13] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, 29(5):1189–1232, 2001. doi: 10.1214/aos/1013203451.

[14] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*, volume 30, pages 3146–3154, 2017. doi: 10.5555/3294996.3295074.

[15] Kewei Cheng, Tao Fan, Yilun Jin, Yang Liu, Tianjian Chen, Dimitrios Papadopoulos, and Qiang Yang. Secureboost: A lossless federated learning framework. *IEEE Intelligent Systems*, 36(6):87–98, 2021. doi: 10.1109/MIS.2021.3082561.

[16] Yang Liu, Yingting Liu, Zhijie Liu, Junbo Zhang, Chuishi Meng, and Yu Zheng. Federated forest. *IEEE Transactions on Big Data*, 8(3):843–854, 2022. doi: 10.1109/TBDATA.2020.2992755.

[17] Lunchen Xie, Jiaqi Liu, Songtao Lu, Tsung-Hui Chang, and Qingjiang Shi. An efficient learning framework for federated XGBoost using secret sharing and distributed optimization. *ACM Transactions on Intelligent Systems and Technology*, 13(5):1–28, 2022. doi: 10.1145/3523061.

[18] Qinbin Li, Zhaomin Wu, Yanzheng Cai, Yuxuan Han, Ching Man Yung, Tianyuan Fu, and Bingsheng He. FedTree: A federated learning system for trees. In *Proceedings of Machine Learning and Systems*, volume 5, pages 89–103, 2023. URL `https://proceedings.mlsys.org/paper_files/paper/2023/hash/3430e7055936cb8e26451ed49fce84a6-Paper-mlsys2023.pdf`.

[19] Qinbin Li, Zeyi Wen, and Bingsheng He. Practical federated gradient boosting decision trees.

In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 4642–4649, 2020. doi: 10.1609/aaai.v34i04.5895.

[20] Qinbin Li, Zhaomin Wu, Zeyi Wen, and Bingsheng He. Privacy-preserving gradient boosting decision trees. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(01):784–791, 2020. doi: 10.1609/aaai.v34i01.5422.

[21] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979. doi: 10.1145/359168.359176.

[22] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, pages 1–10, 1988. doi: 10.1145/62212.62213.

[23] Rosario Gennaro, Michael O Rabin, and Tal Rabin. Simplified vss and fast-track multiparty computations with applications to threshold cryptography. In *Proceedings of the Seventeenth Annual ACM Symposium on Principles of Distributed Computing*, pages 101–111, 1998. doi: 10.1145/277697.277716.

[24] Dheeru Dua and Casey Graff. UCI machine learning repository. `https://archive.ics.uci.edu/ml`, 2019.

[25] Robert Detrano, Andras Janosi, Walter Steinbrunn, Matthias Pfisterer, Johann-Jakob Schmid, Sarbjit Sandhu, Kern H Guppy, Stella Lee, and Victor Froelicher. International application of a new probability algorithm for the diagnosis of coronary artery disease. *American Journal of Cardiology*, 64(5):304–310, 1989. doi: 10.1016/0002-9149(89)90524-9.

[26] W Nick Street, William H Wolberg, and Olvi L Mangasarian. Nuclear feature extraction for breast tumor diagnosis. In *IS&T/SPIE's Symposium on Electronic Imaging: Science and Technology*, volume 1905, pages 861–870. SPIE, 1993. doi: 10.1117/12.148698.

[27] Jack W Smith, James E Everhart, WC Dickson, William C Knowler, and Robert S Johannes. Using the adap learning algorithm to forecast the onset of diabetes mellitus. In *Proceedings of the Annual Symposium on Computer Application in Medical Care*, pages 261–265. American Medical Informatics Association, 1988. URL `https://pmc.ncbi.nlm.nih.gov/articles/PMC2245318/`.

[28] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3–4):211–407, 2014. doi: 10.1561/0400000042.

[29] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 308–318, 2016. doi: 10.1145/2976749.2978318.

[30] Latanya Sweeney. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(05):557–570, 2002. doi: 10.1142/S0218488502001648.