FedBGS: A Blockchain Approach to Segment Gossip Learning in Decentralized Systems

Fabio Turazza, Marcello Pietri, Marco Picone, Marco Mamei
Department of Sciences and Methods for Engineering
University of Modena and Reggio Emilia, Reggio Emilia, Italy
name.surname@unimore.it

***Abstract*—Privacy-Preserving Federated Learning (PPFL) is a Decentralized machine learning paradigm that enables multiple participants to collaboratively train a global model without sharing their data with the integration of cryptographic and privacy-based techniques to enhance the security of the global system. This privacy-oriented approach makes PPFL a highly suitable solution for training shared models in sectors where data privacy is a critical concern. In traditional FL, local models are trained on edge devices, and only model updates are shared with a central server, which aggregates them to improve the global model. However, despite the presence of the aforementioned privacy techniques, in the classical Federated structure, the issue of the server as a single-point-of-failure remains, leading to limitations both in terms of security and scalability. This paper introduces FedBGS, a fully Decentralized Blockchain-based framework that leverages Segmented Gossip Learning through Federated Analytics. The proposed system aims to optimize blockchain usage while providing comprehensive protection against all types of attacks, ensuring both privacy, security and non-IID data handling in Federated environments.**

***Index Terms*—Federated Learning, Gossip Learning, Blockchain, Scalability, Data Privacy, Analytics**

I. Introduction

Over the past decade, owing to the rapid proliferation of digital applications coupled with more affordable and easily implementable data storage solutions, data have acquired substantial legal and economic value. Legally, much of this information is personal and is deemed an inviolable asset for each individual, protected by measures such as the EU GDPR (General Data Protection Regulation) and US HIPAA (Health Insurance Portability and Accountability Act) for sensitive medical information. Economically, companies regard these data as a key asset and are often reluctant to share them. This reluctance creates a sort of "prisoner" dilemma, where data exchange could offer a collective benefit by improving training for a shared model, improving its generalization and reducing biases.

Federated learning was introduced by Google in 2016 through the FedAvg (Federated Averaging) algorithm [1], designed to address these issues by establishing a decentralized training process in which data remain with its original owner. In the FedAvg approach, every participant trains a local model and sends only the model updates to a central server. This server aggregates the updates averaging them, to produce a global model, which is then redistributed to the participants. This cycle is repeated over a predefinite number of rounds until the model converges. Despite ensuring that data stays within each owner's domain, FedAvg has shown vulnerabilities on the security front, as it can be exposed to various types of attacks such as reconstruction attacks, poisoning attacks, and Sybil attacks. Additionally, FedAvg is sensitive to non-iid heterogeneous data from clients, which can lead to notable performance drops.

To address these challenges, Privacy-Preserving Federated Learning (PPFL) [2] was developed with the aim of integrating extra privacy and encryption techniques to enhance the intrinsic security of traditional federated learning. The most widely adopted privacy-preserving method is differential privacy (DP) [3] [4], which works by adding noise (typically Gaussian) to the data to mitigate potential damage from data leaks. Other strategies involve combining FedAvg with methods like SMPC [5] and homomorphic encryption (HE) [6] [7]. Another significant issue highlighted by FedAvg concerns the management of non-identically distributed data among clients: in the presence of heterogeneity in data distribution among participants, the performance of the global model declines exponentially. Several recent studies aim to address the problem of non-IID data in a federated context: FedProx [8] addresses the issue of non-IID data by adding a proximity term to the local loss function, which limits how much client models can deviate from the global model, thereby stabilizing training. SCAFFOLD [9] introduces a correction for client drift through the use of variance reduction in gradients, ensuring more consistent updates among clients and accelerating convergence. MOON [10], on the other hand, leverages contrastive learning to align local models with the global model, reducing differences between local data distributions and improving generalization. FedDyn [11] employs dynamic regularization that balances updates between clients and the central model, compensating for the bias introduced by the heterogeneous data distribution.

To mitigate the issue of heterogeneous data distribution among participants in a federated context, while ensuring the system remains scalable and transparent, we propose FedBGS: a blockchain-based self-segmented gossip learning framework designed to be scalable, model-agnostic, and resilient to non-IID data distributions. FedBGS is compatible with any Ethereum-based blockchain and, by leveraging IPFS, enables optimized data storage that would otherwise be unfeasible on public blockchains.

The main contributions of this work are as follows:

- Federated K-Means++ for Automatic Segmentation: This work introduces a fully decentralized clustering mechanism using federated analytics, enabling automatic segmentation of clients based on data distributions and mitigating the impact of non-IID data.
- Blockchain-based Segmented Gossip Learning Approach: A blockchain-based segmented gossip learning approach is proposed, where clients exchange model updates within dynamically formed segments, ensuring scalability, robustness, and decentralization.
- Hybrid Blockchain and IPFS for Efficient Storage: The system integrates on-chain aggregation with off-chain storage via IPFS, overcoming blockchain scalability issues while maintaining transparency and security in decentralized learning.
- Empirical Validation on Non-IID Data: Extensive experiments on *cifar-10* [12] and *mnist* [13] [14] [15] [16] datasets demonstrate improved security and accuracy compared to traditional federated learning and gossip learning methods in heterogeneous environments and resource-constrained blockchains. Furthermore, theoretical evaluations (Table I) will also be performed on FedBGS complete decentralization and scalability, which represent its true strong point compared to state-of-the-art methods.

II. Related Works

Despite the distributed nature of training, traditional federated architectures rely on centralized aggregation, making the server a critical and vulnerable point for the entire system. Several recent studies have focused on eliminating the server as a single point of failure (SPOF) in favor of peer-to-peer architectures, where aggregation occurs at the client level or via smart contracts leveraging blockchain technology. Frameworks such as FLchain [17], BFLC [18], and BlockFL [19] use blockchain to ensure full decentralization in federated learning, providing participants with complete transparency over aggregation operations and the overall system context. However, similar to traditional federated learning, these frameworks still suffer from performance degradation in the presence of non-IID data, as well as scalability issues when using public blockchains, which are unsuitable for storing large amounts of model parameters. To address the storage limitations inherent in blockchain systems, [20] proposed a hybrid architecture that combines on-chain and off-chain storage solutions. In this approach, the InterPlanetary File System (IPFS) [21] [22] [23] is utilized to store intermediate model parameters, thus mitigating the storage constraints imposed by blockchain bottlenecks.

Gossip learning [24] is a decentralized federated approach in which participants perform on-edge aggregation through a peer-to-peer mechanism, directly exchanging their model weights with other nodes in the network. This methodology overcomes the limitations of a central server, making the system more fault-tolerant and potentially more scalable. Empirical studies have demonstrated that gossip learning can achieve comparable, and in some cases superior, performance to federated learning, particularly when training data is evenly distributed across nodes. Notably [25] provide an in-depth empirical comparison between gossip learning and federated learning, highlighting scenarios where the decentralized approach performs pairly or even exceeds centralized methods.

Split learning [26] [27] is a technique that reduces communication overhead by partitioning the neural network between the client and a central server. In this approach, clients compute the early layers locally and transmit only intermediate activations to the server, which then completes the forward and backward propagation. This strategy minimizes data exchange and enhances privacy by keeping raw data on the client; however, its reliance on a central server can reintroduce centralization issues and potentially limit scalability and security.

A significant extension of gossip learning is segmented gossip learning, which addresses the challenges posed by non-IID data distribution and communication overhead among nodes. Instead of sharing and updating the entire model at each step, only specific segments of the model parameters are exchanged among nodes. [28] discusses how to segment the model, typically the final layer, across clusters of nodes in a decentralized gossip learning framework. This approach reduces communication overhead and mitigates gradient conflicts in heterogeneous data scenarios, leading to faster convergence and improved model accuracy while maintaining a decentralized learning process. In GossipFL [29] , there is not an explicit segmentation but overhead is reduced by transmitting only the most significant model updates through sparsification, which minimizes the data exchanged among nodes. Additionally, adaptive communication dynamically adjusts the frequency and volume of updates based on network conditions and convergence flow, granting efficient resource utilization and faster training.

III. The Proposed Framework

FedBGS (Federated Blockchain Gossip Segmented) is a gossip learning based framework in which peer models are automatically segmented via Blockchain-Scheduling through One-Shot Federated K-means. This segmentation helps protect the system when client data distributions are heterogeneous and to avoid computational overhead. This approach allows handling non-iid data distributions while still providing each component with an overall view of the system. To ensure total transparency and trust among the parties involved (particularly relevant in cross-silo scenarios), FedBGS integrates a blockchain system based on Ethereum smart contracts. This makes it possible to validate participants, preventing malicious clients from infiltrating the process, and makes the system resistant to so-called “poisoning attacks” through dual validation at both the peer and blockchain levels. During the clustering phase, the blockchain also acts as a central aggregator, enabling the system to be fully decentralized and avoiding dependency on a single server that could become a SPOF. The limitations inherent in using Ethereum - a standard blockchain that is highly secure but offers limited scalability,

TABLE I: Comparison of FedBGS vs. Federated Learning (FL) and Blockchain-FL Approaches

Approach	Non-IID Data Handling	Hybrid Storage (Blockchain + IPFS)	Privacy-Preserving Techniques	Scalability	Full Decentralization
FedAvg [1]	Limited, requires enhancements	×	Basic privacy, vulnerable to inference attacks	High, but centralized	×
FLchain [17]	Moderate, inherits FedAvg limitations	Partial (On-chain only)	Transparency-focused, no additional privacy (DP, HE)	Moderate, consensus bottleneck	✓
BFLC [18]	Limited, no special handling	Limited, On-chain with committee consensus	Robustness via committee validation	Improved but limited at scale	✓
BlockFL [19]	No special handling, standard FedAvg	On-chain only, high storage overhead	Transparent validation, no DP or encryption	Limited, consensus overhead	✓
BGFL [30]	Limited explicit non-IID handling, focuses on gossiping	On-chain only, no IPFS	Decentralized, no explicit DP or encryption	High via gossip communication	✓
FedBGS (Proposed)	**Robust (One-Shot K-Means++ clustering)**	**Hybrid (Blockchain + IPFS)**	**HE, DP and Trimmed-Mean Aggregation**	**High, decentralized segmented training**	✓

Note: The bold entries indicate the notable strengths of FedBGS compared to other existing approaches.

is inefficient for on-chain storage of model weights, and has high gas fees - are addressed by a hybrid approach in which aggregation and storage are handled off-chain via IPFS, while a reference to the completed transaction is saved on-chain, along with the entire history of the federated process.

A. Phase 1: Auto-Clustering through Federated Analytics (1)

With the emergence of federated learning, another decentralized and privacy-oriented approach named Federated Analytics (FA) [31], began to take shape, of which federated learning is a subcategory. FA differs from classical federated learning because it is not aimed at training a global AI model, but rather at performing simple statistical calculations in a federated manner. Consequently, it has a much simpler structure and does not rely on multiple rounds. One-Shot Federated K-Means is a specific case of Federated Analytics in which the centroids are derived from the peers in a single round of communication. This makes it possible to group peers into clusters based on their similarity, without revealing each participant's internal data. As in traditional federated structures, federated analytics also depends on the presence of a central aggregator. The advantage of this approach, however, lies in the fact that it does not require the sharing of full deep learning models. Simply sending statistical parameters, such as centroids, which require vastly less storage than weights, enables the use of smart contracts for on-chain storage and aggregation without incurring excessive fees. It is worth noting that the clustering step was executed using a one-shot method typical of federated analytics, chosen specifically to avoid overloading the blockchain. On the flip side, one-shot methods often compromise clustering accuracy, typically settling for suboptimal solutions. However, our experiments revealed that this slight performance drop is generally negligible, even when employing a suboptimal approach. That said, few-shot iterative methods, which fall under the federated learning umbrella, can still be considered a viable option if their blockchain storage trade-offs are carefully assessed. To boost the overall precision of the clustering process, we employed the K-Means++ algorithm [32], which selects the first centroid randomly and then chooses subsequent centroids based on a probability distribution proportional to the squared distance from the nearest existing centroid. This approach reduces randomness and improves clustering efficiency. FedBGS will use the clusters generated in the initial phase to create segments of the global model; each cluster corresponds to a segment derived from the model, meaning each cluster trains a different part of the model. The segmentation is done by splitting the model's last layer into equal parts, ensuring that each cluster trains a specific portion of the model. This approach reduces gradient conflicts as well as computational and communication overhead.

The retrieval process works by accessing an on-chain mapping that associates each cluster with a specific range of neurons in the last fully connected layer. When a peer requests its segment boundaries, the smart contract fetches the assigned start and end indices from storage and returns them. This ensures that each cluster updates only the designated part of the model, preventing unauthorized modifications and enforcing decentralized segmentation.

Let C_k be the k-th cluster and $\mathcal{S}_k$ denote its assigned segment of the model, defined by the neuron index range $[s_k, e_k]$. For a model parameter w_i, the update rule enforced by the smart contract is:

$$\mathcal{S}_k = \{w_i \mid s_k \leq i \leq e_k\} \tag{1}$$

$$\forall w_i \in \mathcal{W}, \quad w_i \text{ is updated only if } w_i \in \mathcal{S}_k \tag{2}$$

Legend:
C_k: The k-th cluster.
$\mathcal{S}_k$: The segment assigned to cluster C_k.
s_k, e_k: Start and end indices defining the segment for C_k.
w_i: A model parameter in the last fully connected layer.
$\mathcal{W}$: The set of all model parameters in the last layer.

B. Blockchain Integration

Ethereum is a popular blockchain platform that lets people build and run smart contracts and decentralized applications without a central authority. The blockchain is made up of blocks, where each block (single block structure is shown in Fig. 1) is a list of transactions and a reference to the previous block. This linking of blocks creates a secure chain that makes it hard to change any past data.

In Ethereum, transactions are verified and new blocks are created by network participants. In the original system, these participants were called miners, and they used a method called Proof-of-Work (PoW) to solve complex puzzles before adding a block. However, Ethereum has now moved to a system called Proof-of-Stake (PoS). In PoS, instead of miners, there are validators who are chosen to create blocks based on the amount of Ether they lock up (or "stake"). This method uses much less energy and is more efficient.

Modern validation techniques in Ethereum focus on making sure that validators act honestly. For example, if a validator behaves badly, the system can "slash" (or penalize) their stake. This helps keep the network secure and reliable.

We chose to use Ethereum for our project on federated gossip learning and as a server for federated one-shot K-means because it offers a decentralized and transparent way to verify updates and client registrations. By using Ethereum, we can ensure that every participant's contribution is recorded in a secure, tamper-proof ledger.

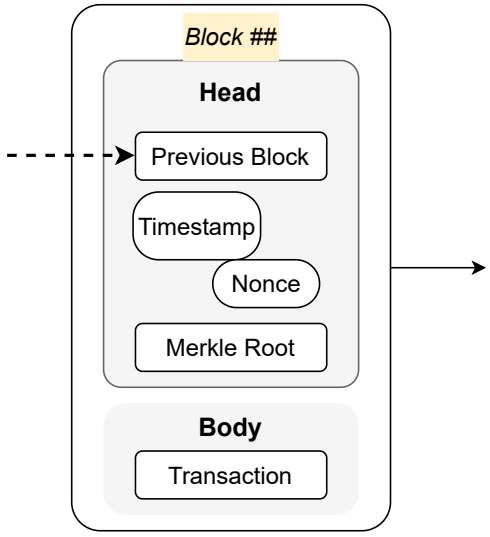


Fig. 1: Example of an Ethereum Block: In Ethereum blocks, the Merkle root [33] is a single hash that summarizes and verifies all transactions in the block, ensuring data integrity. The nonce is a number used in the consensus process to achieve a valid block hash. The previous block hash links the current block to the one before it, creating an immutable chain.

In Decentralized learning, blockchain is typically used for the following purposes:

- **Decentralization:** Decentralizing the system by performing on-chain aggregation via smart contracts, thus eliminating the server as a single point of failure.
- **Immutable Client Registry:** Maintaining an immutable ledger of clients authorized to participate, which helps prevent Sybil attacks.
- **Secure Transaction Recording:** Ensuring that transactions (i.e., updates sent by nodes) are recorded in a distributed and unchangeable manner, thereby reducing the risk of tampering (poisoning attacks).
- **Double Spending Prevention:** Preventing double spending attacks, where a node might try to reuse old updates or replicate them as new to manipulate the federated model, since every transaction is uniquely tracked and cannot be "spent" again.
- **Incentivization:** Implementing incentive systems through token-based smart contracts.

C. IPFS and Blockchain

IPFS (InterPlanetary File System) is a system for storing and sharing files in a distributed manner, without relying on centralized servers. The basic principles of IPFS are as follows:

- **Content Addressing:**
  In IPFS, files are not identified by their location (such as a URL), but by their content as show in Fig. 2. When you add a file, it is transformed into a unique cryptographic hash. This hash becomes the *address* of the file, ensuring that each file is securely and immutably identified.
- **Blocks and the Merkle DAG:**
  A file in IPFS is divided into small pieces called *blocks*. Each block contains:
  - **Data:** A portion of the file.
  - **Links:** References to other blocks (if the file is split into multiple parts).

  These links form a structure called a *Merkle DAG* (Directed Acyclic Graph), which allows verification that each block has not been altered. Even if a single block changes, the overall hash will change, indicating a modification.
- **Distribution:**
  When you request a file by its hash, IPFS searches the entire network for the block (or blocks) corresponding to that hash and reassembles them to reconstruct the file.

Both IPFS and Ethereum utilize a Merkle graph structure to efficiently ensure data integrity and verify content with minimal cryptographic proofs. In Ethereum, this design (Patricia-Tree) enables fast, decentralized verification of transactions and state changes, reducing gas costs and enhancing scalability. Similarly, IPFS's use of a Merkle graph allows for immutable, content-addressed storage, creating a natural synergy that makes integration between the two systems both seamless and robust.

With FedBGS, the use of the blockchain will be divided into two phases:

Phase 1:

- **Task 1:** The blockchain will have the role of managing the registration of each participant through unique credentials; an unregistered client will not have the right to participate in the shared training.
- **Task 2:** We will perform the aggregation of the centroids for the federated k-means via smart contracts and store the results directly on-chain.

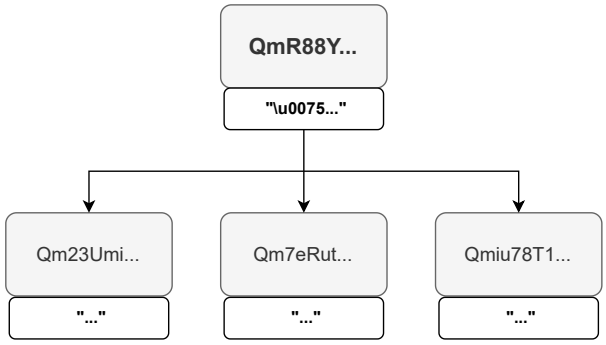


Fig. 2: The block size in IPFS can vary depending on the stored data. Thanks to a hierarchical structure based on hash links, each block points to other linked blocks, promoting redundancy and sharing across nodes. This scalable approach ensures fast and reliable retrieval of content from various locations.

- **Task 3:** The smart contract, using an internal policy algorithm, determines which part of the model will be assigned to each cluster. At the start of the gossip cycle, the client queries the smart contract to learn its designated cluster and which "segment" of the model it should train.

Phase 2:

- **Task 1:** At regular intervals, the smart contract will randomly select a leader to handle global inter-segment aggregation by regenerating the original, "non-segmented" model, which is then stored on IPFS.
- **Task 2:** When each peer retrieves the global model, a validation process is carried out to verify the model's integrity (the hash of the model extracted by the peer must match the immutable one stored on the blockchain).
- **Task 3:** A token-based incentive system rewards clients who provide a better model to the subsequent client, penalizing those who deviate too much from the current loss.

D. Privacy-Preserving Decentralized Learning

As mentioned in the introduction, the intrinsic security level provided by gossip learning is not sufficient to ensure an adequate level of privacy, especially in cross-silo contexts. The introduction of blockchain with IPFS into the system aims to prevent some possible attacks such as Sybil or poisoning attacks; data leakage or the potential espionage by "curious" participants remains an unresolved issue with the previously described technologies, particularly during the clustering phase, where the transmission of statistical data such as centroids rather than parameters of a deep model makes the system especially vulnerable to several types of attacks, including:

- **Membership Inference Attacks:** An adversary attempts to determine whether a particular sample (e.g., a user's data) was present in the training set.
- **Gradient Inversion (or Reconstruction) Attacks:** The attacker tries to reconstruct sensitive original data (images, text, etc.) from the gradients shared by clients during training.
- **Model Inversion Attacks:** The goal is to infer sensitive attributes of the training examples by leveraging information returned by the model.
- **Eavesdropping / Man-in-the-Middle Attacks:** An attacker intercepts or modifies parameters (or gradients) while they are transmitted between participants and the central server (or among peers).
- **Model Extraction Attacks:** The adversary attempts to extract or replicate the model's architecture and parameters by making queries or analyzing the system's responses.

In FedBGS, we integrate several advanced techniques to enhance the security and robustness of our decentralized gossip learning system, which leverages blockchain and IPFS. These techniques, including Differential Privacy, Homomorphic Encryption, and robust aggregation via the trimmed mean, work together to protect sensitive information, ensure secure aggregation, and defend against adversarial attacks.

Differential Privacy (DP): DP ensures that the contribution of each client remains private, even when model updates are shared. In federated settings, *local DP* involves adding noise at the client level before sending updates, whereas *global DP* applies noise during the aggregation process. Within our gossip learning framework, DP is used during the training phase to clip and perturb gradients, thereby safeguarding individual data contributions while still enabling effective collaborative learning.

During the training phase in segmented gossip learning, each client's gradient g is made differentially private by clipping and adding Gaussian noise. This is represented by:

$$\tilde{g} = g \cdot \min\left(1, \frac{C}{\|g\|_2}\right) + \mathcal{N}\left(0, \sigma^2 C^2\right) \tag{3}$$

Legend:
g: Original gradient computed by the client.
$\tilde{g}$: Differentially private gradient.
C: Clipping threshold.
$\|g\|_2$: L_2-norm of the gradient g.
$\mathcal{N}(0, \sigma^2 C^2)$: Gaussian noise with mean 0 and variance $\sigma^2 C^2$.
σ: Noise multiplier controlling the amount of noise added.

FedBGS implements an internal scheduler that allows DP to be changed dynamically. Our scheduler applies a linear decrease to the DP over time, reaching the minimum level in the final iteration. The early rounds are particularly delicate because the model is directly influenced by each node's initial data. Since the model has not yet been "mixed" with information from many other participants, the updates sent may contain more identifiable traces of the original data. In other words, privacy could be at greater risk because the "noise" (i.e. the variability introduced by other nodes and any artificial DP noise) has not yet accumulated enough to mask the individual characteristics of the local datasets.

Furthermore, in the initial rounds, if the updates are more significant, it is necessary to introduce more noise to keep them protected, which can quickly consume the privacy budget. If this budget is exhausted or significantly reduced in the very early rounds, there is a risk of not being able to guarantee the same level of protection in subsequent iterations.

Homomorphic Encryption (HE): HE allows computations to be carried out on encrypted data without revealing the underlying information. Although *fully homomorphic encryption* supports arbitrary computations on ciphertexts, its high computational cost makes it less practical. Instead, we employ *partial homomorphic encryption*, specifically, the additively homomorphic Paillier scheme to securely aggregate label distributions and model updates. This enables us to combine encrypted client updates without exposing their raw values, maintaining both privacy and integrity during the aggregation process.

To protect the centroids during the clustering phase, we aggregate the encrypted label distributions using the Paillier scheme [34]. Let $x_{i,j}$ be the j-th component of the label distribution (centroid) from client i. With $E(\cdot)$ and $D(\cdot)$ denoting the encryption and decryption functions, respectively, and using the homomorphic addition operator $\bigoplus$, the aggregated (and decrypted) centroid component is computed as:

$$\hat{c}_j = \frac{D(\bigoplus_{i=1}^{n} E\left(x_{i,j}\right))}{n} \tag{4}$$

Legend:
$E(x)$: Encryption of x using the Paillier scheme.
$D(y)$: Decryption of ciphertext y.
$\bigoplus$: Homomorphic addition operator.
n: Total number of clients (or label distributions).
$x_{i,j}$: j-th component of client i's label distribution.

In **FedBGS**, partial homomorphic encryption is used only during the clustering phase, which we identify as the most vulnerable from a security standpoint. It is not applied throughout the entire process to avoid the computational overhead caused by gossiping (more data to encrypt and more iterations).

Trimmed Mean Aggregation: To counteract Byzantine attacks [35], where some nodes might submit malicious updates, aggregation is computed utilizing the trimmed mean. This robust statistical method discards a fixed proportion of the most extreme values before averaging, thereby reducing the impact of outliers. In the context of our gossip learning system, the trimmed mean helps maintain the reliability of the global model by mitigating the effects of adversarial contributions.

Together, these techniques enable our system to perform secure, privacy-preserving, and robust model aggregation in a federated environment. By combining DP for privacy, partial HE for secure computations, and trimmed mean for robust aggregation, our approach is well-suited to address the unique challenges posed by decentralized and potentially adversarial settings.

Let $\{\theta_i\}_{i=1}^{n}$ be the set of model updates and $\theta_{(i)}$ denote the i-th sorted update. With a trim ratio r, the robust aggregated update θ^* is computed as:

$$\theta^* = \frac{1}{n(1-2r)} \sum_{i=\lceil rn \rceil}^{\lfloor (1-r)n \rfloor} \theta_{(i)} \tag{5}$$

Legend:
θ_i: The i-th individual model update.
$\theta_{(i)}$: The i-th order statistic (sorted update).
n: Total number of updates.
r: Trim ratio (fraction of values trimmed from each end).
θ^*: The robust aggregated model update.

Algorithm 1 FedBGS Phase 1: On-chain Clustering & Client Registration

Require: Peer set P with local data D_p
Ensure: Each peer $p \in P$ is registered on-chain and assigned to a segment S_i

for each peer $p \in P$ **do**
  Register p on the blockchain with unique credentials
  Compute local label distribution x_p from D_p
end for
Form segments $S_1, S_2, \ldots, S_S$ via federated k-means++ on $\{x_p\}_{p \in P}$
for each segment S_i **do**
  Secure Aggregation: Aggregate label distributions using Partial HE ▷ see HE formulation in Eq. (4)
  Compute secure centroid $\hat{c}_i$ for segment S_i
  Update $\hat{c}_i$ on-chain (acting as the aggregation server)
end for
Cluster Discovery: Each peer $p \in P$ retrieves its assigned segment
  S_i via smart-contract query subject to Eq. (1)

E. Phase 2: Segmented Gossip Learning (2)

As mentioned earlier, gossip learning (GL) is a strategy that allows training a shared model in a completely decentralized manner, without participants explicitly sharing their data. Gossip learning differs from federated learning (which is semi-decentralized) in that it does not rely on a central server for aggregation; instead, aggregation is performed at the client level, making the system immune to malicious or faulty servers.

Gossip learning can operate under two different paradigms:

Synchronous Logic: This approach retains the concept of rounds as in federated learning, where all peers wait for every participant to complete the training phase before sending their model updates. Although this method is simpler and more orderly from an analytical standpoint, it creates significant bottlenecks when clients are heterogeneous, making the system less scalable in scenarios with many participants having varying computational capabilities and data. Asynchronous Logic: In this approach there are no rounds; each client aggregates model updates at different times based on its internal policies.

Algorithm 2 FedBGS Phase 2: Segmented Gossip Learning with Hybrid Storage & Global Aggregation

Require: Peer set P with assigned segments S_i, DP parameters C, σ, trim ratio r
Ensure: Global aggregated model parameters θ^*

for each peer $p \in P$ within its assigned segment S_i **do**
  Train local model and compute gradient g_p
  Apply Differential Privacy: Clip and perturb g_p to obtain $\tilde{g}_p$ ▷ see DP formulation in Eq. (3)
  Share $\tilde{g}_p$ via hybrid storage
  – Store model update on IPFS
  – Record update hash on-chain for integrity verification
  Validate the update by matching the IPFS hash with the blockchain record
  if validation fails **then**
    Penalize p via smart-contract enforced token deduction
  end if
end for
Global Aggregation:
Elect a leader p^* via smart contract (e.g., using on-chain randomness)
for each segment S_i **do**
  Leader p^* collects updates $\{\tilde{g}_p\}_{p \in S_i}$
  Aggregate updates using trimmed mean (Eq. (5))
  Obtain segment update $\Delta\theta_i$
  Update global model segment i with $\Delta\theta_i$
end for
Leader p^* reconstructs the complete global model θ^*
Store θ^* on IPFS and record its CID on-chain
Each peer $p \in P$ retrieves the latest CID and updates its local model

While this strategy is less orderly, it offers better performance compared to the synchronous method.

As outlined in the overall architecture shown in Fig. 4, FedBGL operates in two phases:

Phase 1 (Fig. 3a): On-Chain Clustering & Client Registration:

- **Client Registration:** Each peer registers on the blockchain using unique credentials.
- **Local Computation:** Every peer computes its own local label distribution from its local dataset.
- **Clustering:** A federated k-means++ algorithm is applied to these distributions to form homogeneous segments.
- **Secure Aggregation:** For each segment, the label distributions are aggregated securely using partial homomorphic encryption to compute a secure centroid.
- **On-Chain Update:** The secure centroids are updated directly on-chain, acting as the aggregation servers.
- **Segment Assignment:** Each peer retrieves its assigned segment S_i by querying the blockchain via the smart contract.

Phase 2 (Fig. 3b): Segmented Gossip Learning with Hybrid Storage & Global Aggregation:

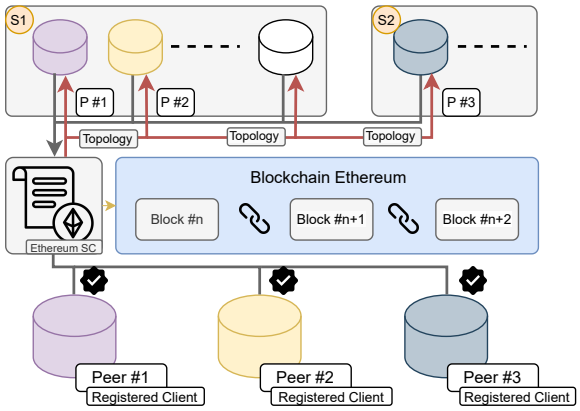
- **Local Training:** Within each segment, peers train their local models and compute gradients.
- **Differential Privacy:** Each gradient is processed with differential privacy mechanisms to protect sensitive information.
- **Hybrid Storage:** The privacy-preserving updates are shared by:
  - Storing the model update on IPFS.
  - Recording the corresponding update hash (CID) on the blockchain for verification.
- **Update Verification:** Each peer validates the update hash by comparing the IPFS hash with the blockchain record. In case of validation failure, the peer is penalized via blockchain token deduction.
- **Leader Election:** A leader peer is elected on-chain via a smart contract (e.g., using on-chain randomness).
- **Robust Aggregation:** The elected leader collects the privacy-preserving updates from peers in each segment and aggregates them using a trimmed mean approach to compute the segment update.
- **Global Model Reconstruction:** The leader reconstructs the complete global model from the aggregated segment updates.
- **Global Storage & Dissemination:**
  - The leader stores the global model on IPFS and records the corresponding CID on-chain.
  - Each peer retrieves the latest global model via the recorded CID and updates its local model segment accordingly.

In FedBGS, in the second phase, we implemented an asynchronous gossip learning system where peers, operating independently (each in its own thread), perform local model training. Training, sharing, and testing operations occur asynchronously, with variable intervals determined randomly.

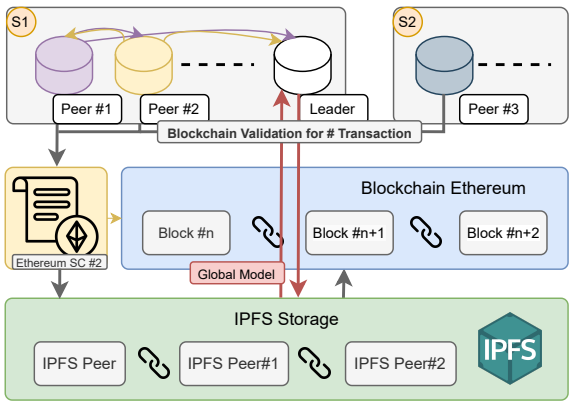
Global Model Derivation and Iteration of Gossip Learning

In the proposed protocol, each peer begins by training its local model on its private dataset and computing a gradient update, which is then processed through Differential Privacy mechanisms to preserve data confidentiality. Within each homogeneous segment, obtained via a federated k-means++ clustering algorithm, peers exchange their privacy-preserving updates with their designated neighbors. The intra-segment aggregation, performed using a robust method such as the trimmed mean, refines the local model for that segment.

At regular intervals, the system elects a leader peer on-chain via a smart contract that employs a randomization mechanism. The elected leader collects the aggregated updates from all segments and fuses them into a unified global model. Subsequently, the leader stores the global model on IPFS and records the corresponding CID on the blockchain. Finally, each



(a) **Phase 1**: Each client is registered on the blockchain and clustered based on data similarity. The blockchain serves as an aggregator for centroid updates, leveraging a federated k-means clustering approach. The FedClustering operates in a decentralized manner using a One-Shot approach, where the centroids are stored on-chain. Cluster assignments directly influence model segmentation, as each cluster is responsible for a subset of the model params.



(b) **Phase 2**: Clients perform differentially private cluster-based segmented gossip learning, where only specific segments of the last fully connected layer are updated by each cluster. Model updates are stored and retrieved from IPFS, ensuring efficient validation and learning transparency. Aggregation on the client side is performed using trimmed-mean aggregation, and the incentive system for peer training is managed through a dedicated smart contract.

Fig. 3: FedBGS overall architecture with cluster-based segmented gossip learning. Thanks to smart contracts and Ethereum's execution logic, every operation delegated to the blockchain is executed across multiple blocks and validated. The segmentation strategy ensures efficient training on non-IID data by limiting updates to assigned neuron blocks in the final layer.

peer retrieves the updated global model using the recorded CID and updates its local model accordingly.

This iterative process of local training, intra-segment aggregation, leader-driven global aggregation, and model dissemination ensures continuous improvement and convergence of the global model.

IV. Experimental Evaluation

The objective of this experimental evaluation is to graphically demonstrate the convergence of FedBGS using various datasets and under different initial data distribution conditions. The contribution of individual participants to the training will be shown through line charts and bar plots, and a table summarizing the impact of FedBGS on an Ethereum blockchain in terms of costs will be included. Other evaluations related to the security and scalability of the system will be conducted at a theoretical level. To validate the performance of FedBGS, we tested it on 6 different standard datasets. This document provides an elegant overview of the standard datasets used to validate the performance of FedBGS. The datasets are grouped into two main categories: those based on grayscale images (28×28) and the CIFAR-10 dataset, which uses color images (32×32). The following experiments were all executed locally on a NVIDIA RTX 4080 laptop GPU and are fully replicable by following the deployment instructions on GitHub[1].

Grayscale Image Datasets (28×28)

These datasets are characterized by their uniform size and grayscale format. They are widely adopted as benchmarks due to their simplicity, standardized structure, and ease of comparison across models.

- **MNIST:**
  Contains handwritten digits (0–9) represented as 28×28 grayscale images. It comprises 60,000 training images and 10,000 test images.
- **EMNIST:**
  An extension of MNIST that includes additional letter classes. The "digits" split mirrors MNIST, offering a larger dataset with approximately 280,000 images in total.
- **Fashion-MNIST (FMNIST):**
  Consists of 28×28 grayscale images of clothing items, with 60,000 training and 10,000 test images. It provides a more challenging alternative to digit recognition.
- **KMNIST:**
  Comprises 28×28 grayscale images of Japanese cursive (Kuzushiji) characters. It generally follows a similar split to MNIST (around 60,000 training and 10,000 test images), offering a unique visual challenge.

CIFAR-10

- **CIFAR-10:**
  Consists of 32×32 RGB color images across 10 classes (e.g., airplanes, cars, birds). It includes 50,000 training images and 10,000 test images, and its natural complexity and diversity make it a challenging benchmark for image classification tasks.

A. Client Models

For the experimental validation of the results, we implemented two different convolutional networks. The first (NetM-

[1]https://github.com/FabioTur-dev/gossip_bc_full

NIST) is used generically for all grayscale datasets derived from MNIST; the second, on the other hand (NetCIFAR), is more complex and is used to achieve the desired performance with an RGB dataset such as CIFAR-10. Further experiments on larger datasets such as STL-10 and ImageNet were not conducted due to our hardware limitations.

NetMNIST

The **NetMNIST** network is a CNN specifically designed for the MNIST dataset, which contains 28×28 grayscale images representing handwritten digits. Its architecture consists of:

- **Convolutional Layers:** Two convolutional layers are used. The first layer employs 6 filters with a kernel size of 5, while the second layer uses 16 filters (also with a kernel size of 5). These layers, followed by ReLU activations and max pooling operations, extract the salient features from the images.
- **Flattening:** The output of the convolutional layers is flattened to prepare it for the subsequent layers.
- **Fully Connected Layers:** A sequence of three fully connected layers is used, with 120 and 84 neurons in the first two layers, respectively, before mapping the data into the 10 classes (from 0 to 9).

This simple and lightweight architecture is ideal for digit recognition due to the standardized and relatively simple nature of the MNIST dataset.

NetCIFAR (for CIFAR-10)

The **NetCIFAR** network was developed for the CIFAR-10 dataset, which consists of 32×32 RGB color images distributed across 10 classes. Its architecture is notably more complex, featuring:

- **Convolutional Structure:** A sequence of convolutional layers with 3×3 kernels is employed. The first block uses 32 filters, the second block uses 64 filters, and the third block utilizes 128 filters. Each convolutional layer is followed by batch normalization and ReLU activations to enhance training stability.
- **Pooling:** Two max pooling operations are applied, reducing the spatial dimensions of the feature maps to 8×8, which facilitates handling the increased image complexity.
- **Classifier:** After flattening the feature maps, the classifier includes a fully connected layer with 256 neurons, followed by dropout to mitigate overfitting, and finally a linear layer that maps the features into the 10 classes.

This higher number of filters and neurons highlights the increased complexity and diversity of the natural images in the CIFAR-10 dataset, ensuring a good balance between generalization capacity and training stability.

Dirichlet Partitioning

In our setup, we partition the dataset among clients using a Dirichlet distribution [36] [37]. This approach simulates non-IID data by assigning different proportions of each class to different clients. For a dataset with K classes and N clients, we generate, for each class k, a probability vector

$$\mathbf{p}_k \sim \text{Dirichlet}(\beta, \beta, \ldots, \beta), \tag{6}$$

where β is a concentration parameter that controls the degree of heterogeneity. A smaller β produces a more imbalanced distribution, whereas a larger β results in a more uniform, IID-like partitioning.

For each class k, given that there are n_k samples, the number of samples allocated to client i is calculated as:

$$n_{k,i} = \lfloor p_{k,i} \cdot n_k \rfloor, \tag{7}$$

where $p_{k,i}$ is the i-th component of $\mathbf{p}_k$. Any remaining samples are then randomly distributed among the clients.

This partitioning strategy is applied to simulate realistic data heterogeneity among clients, which is critical for evaluating the robustness and performance of our federated learning approach under non-IID conditions.

In the three accuracy graphs reported in the Figure 4, it is shown that as the β parameter varies (the smaller the value of β, the more heterogeneous the distributions among peers are), the overall training quality remains almost unchanged. The same is not the case, for instance, with FedAvg, which suffers a significant drop in performance as β decreases. What we want to demonstrate with these results is that training performance comparable to state-of-the-art methods can be achieved with a fully decentralized automated system, without trade-offs in terms of scalability, costs, or security. The use of a public Ethereum-based blockchain is one of the key elements in this decentralization, a feature that could not be achieved with a private blockchain since it is still managed by a single entity. The bar plots more clearly show the alignment of the various peers in terms of accuracy; the accuracy achieved, in itself, is not important in this experiment, as it is strictly influenced by the available hardware resources and the use case, namely the network used with its respective hyperparameters and the privacy budget required to ensure a sufficient level of system security, a budget that is closely tied to the dataset and the amount of data available. The parameter by which we will evaluate the effectiveness of the system will be the accuracy growth delta, in addition to a direct theoretical comparison with state-of-the-art systems on the key aspects of FedBGS. Another important aspect highlighted by the bar plots concerns the convergence speed of individual clients' alignment: the black markers indicated in each plot represent the accuracy history for each gossip iteration, and their sparsity in cases with lower β values signifies a slower convergence speed, as well as a generally lower accuracy. Resolving the trade-offs between privacy, scalability during clustering, and accuracy will be crucial in the two design phases of FedBGS, depending on the size and type of datasets involved. Figure 5, on the other hand, shows the accuracy values recorded on each individual peer for the cifar-10 dataset. Despite the classification challenges posed by an RGB dataset compared to the MNIST variants, the model proves to be very robust during convergence, even in the presence of non-IID data.

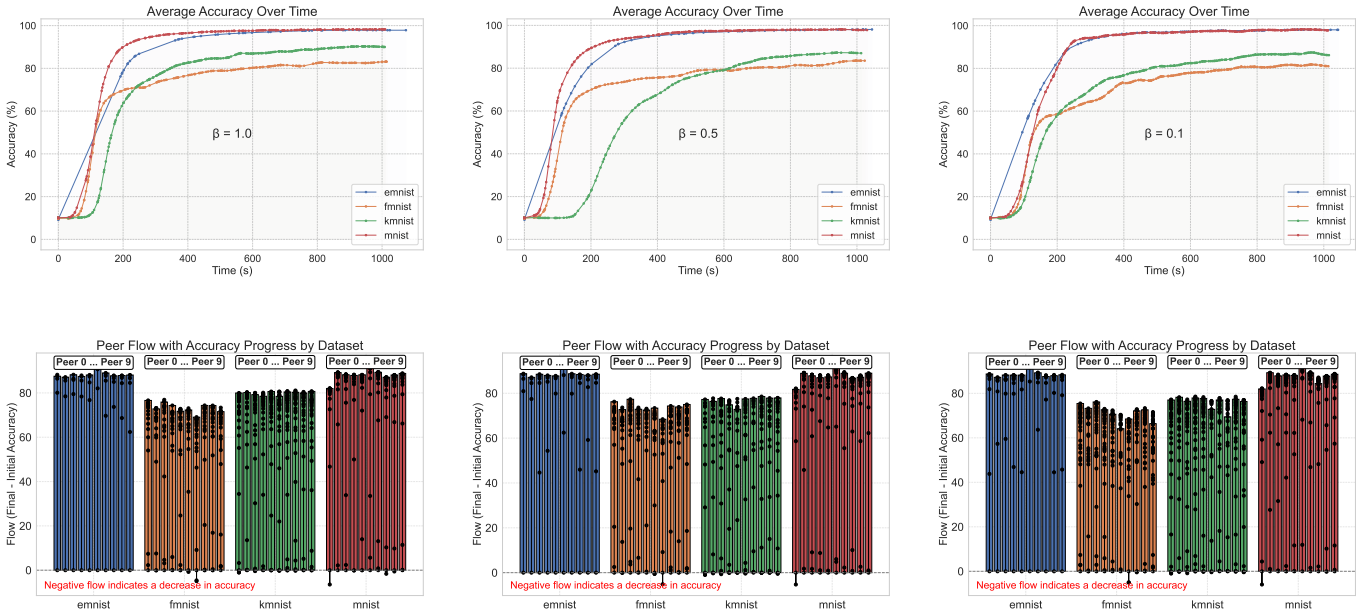


Fig. 4: In the three line charts above, the average accuracies recorded on all grayscale datasets (MNIST and variants) are reported; from left to right, the results correspond to β=1.0, β=0.5, and β=0.1 (the latter under conditions of high heterogeneity among peers). In the lower section, the respective bar plots are shown (each bar plot corresponds to the line chart above it), representing the final and intermediate values for each individual peer.

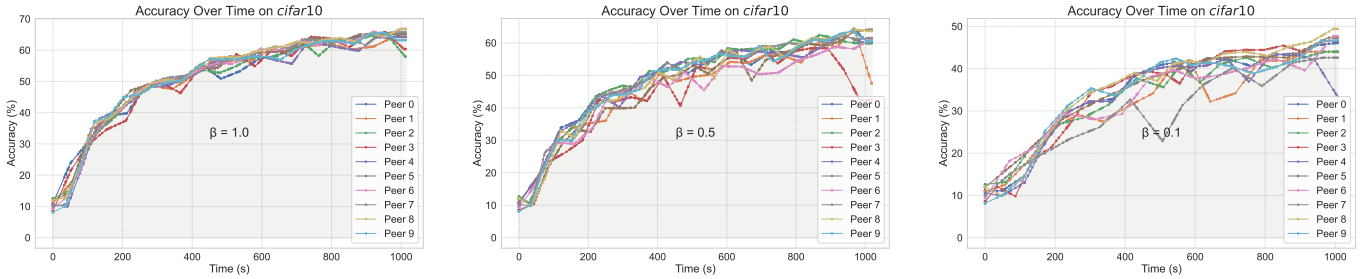


Fig. 5: Accuracy trend for each peer on the *CIFAR-10* dataset. As in the previous line charts, performances with three different β values (1.0, 0.5, 0.1) are highlighted to demonstrate the robustness of FedBGS when using more complex RGB datasets.

TABLE II: FedBGS Blockchain Usage with Segmented Gossip Learning

Operation Type	**Gas Usage (Gas Unit)**[a]		
	Interaction Cost	***Times***	***SC Type***
SC Deployment	1.418.084	Once	#1
SC#2 Deployment	1.566.634	Once	#2
Registration	100.340	for Each Client	#1
Reset Balance (Optional)	257.032	for Each Client	#2
Token Penalization	77.102	Variable	#2
Save Hash	50.527	for Each Client	#2
Save Cluster Centers	257.000	Once per Update	#1
Assign Segment to Peer	120.450	for Each Client	#1
Retrieve Segment Boundaries	35.210	Each Training Round	#1
Validate Segment Update	65.800	Variable (Each Update)	#2

[a]Note that the gas usage also depends on the smart contract's optimization.

In the table II, we show the Ethereum gas used by each function involved in both phases of FedBGS. The functions are split into two types of smart contracts as illustrated in the figure 4: #1 is dedicated to registration and clustering functions, while #2 is used for the gossiping and penalty phase. While our native choice for FedBGS was Ethereum, as it represents a standard providing high levels of security and functionalities, our gossiping system is designed to support any blockchain that offers advanced capabilities for dApps, such as Cardano, Solana, or PolkaDot. Hence, any blockchain of this type can be a viable choice for implementing FedBGS, with the selection depending on the specific requirements of one's system.

V. Conclusions

To conclude, it can be stated that FedBGS achieved excellent results in terms of convergence by proposing a robust

decentralized learning method, while addressing several open challenges in federated learning including full decentralization and the management of heterogeneously distributed data among participants. Furthermore, full decentralization is of fundamental importance given users' reluctance to trust a centralized structure, and many state-of-the-art methods that propose "serverless" architectures present significant trade-offs in terms of scalability (a problem encountered in many blockchain-based approaches). Future developments of FedBGS will focus on limiting the impact of local differential privacy which, in the presence of few data, significantly degrades overall performance despite scheduling and on methods to adapt FedBGS to constrained architectures, a challenge that proves even more demanding in the context of gossip learning due to the absence of a server.

ACKNOWLEDGMENT

This research has been supported by the project "A catalyst for EuropeaN ClOUd Services in the era of data spaces, high-performance and edge computing(NOUS)", Grant Agreement Number 101135927.

REFERENCES

[1] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," 2023. [Online]. Available: https://arxiv.org/abs/1602.05629

[2] X. Yin, Y. Zhu, and J. Hu, "A comprehensive survey of privacy-preserving federated learning," *ACM Computing Surveys (CSUR)*, vol. 54, pp. 1 – 36, 2021. [Online]. Available: https://api.semanticscholar.org/CorpusID:236898578

[3] Z. Ji, Z. C. Lipton, and C. Elkan, "Differential privacy and machine learning: a survey and review," 2014. [Online]. Available: https://arxiv.org/abs/1412.7584

[4] A. E. Ouadrhiri and A. Abdelhadi, "Differential privacy for deep and federated learning: A survey," *IEEE Access*, vol. 10, 2022.

[5] S. M. Hosseini, M. Sikaroudi, M. Babaei, and H. R. Tizhoosh, "Cluster based secure multi-party computation in federated learning for histopathology images," 2022. [Online]. Available: https://arxiv.org/abs/2208.10919

[6] W. Jin, Y. Yao, S. Han, J. Gu, C. Joe-Wong, S. Ravi, S. Avestimehr, and C. He, "Fedml-he: An efficient homomorphic-encryption-based privacy-preserving federated learning system," 2024. [Online]. Available: https://arxiv.org/abs/2303.10837

[7] A. Viand, C. Knabenhans, and A. Hithnawi, "Verifiable fully homomorphic encryption," 2023. [Online]. Available: https://arxiv.org/abs/2301.07041

[8] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," 2020. [Online]. Available: https://arxiv.org/abs/1812.06127

[9] S. P. Karimireddy, S. Kale, M. Mohri, S. J. Reddi, S. U. Stich, and A. T. Suresh, "Scaffold: Stochastic controlled averaging for federated learning," 2021. [Online]. Available: https://arxiv.org/abs/1910.06378

[10] Q. Li, B. He, and D. Song, "Model-contrastive federated learning," 2021. [Online]. Available: https://arxiv.org/abs/2103.16257

[11] D. A. E. Acar, Y. Zhao, R. M. Navarro, M. Mattina, P. N. Whatmough, and V. Saligrama, "Federated learning based on dynamic regularization," 2021. [Online]. Available: https://arxiv.org/abs/2111.04263

[12] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[13] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.

[14] A. Krizhevsky, "Learning multiple layers of features from tiny images," University of Toronto, Tech. Rep., 2009.

[15] G. Clanuwat, H. Kuratani, and Y. Wada, "Kuzushiji-mnist: a novel benchmark dataset for modern japanese (kuzushiji) recognition," 2018.

[16] G. Cohen, S. Afshar, J. Tapson, and A. van Schaik, "Emnist: an extension of mnist to handwritten letters," in *2017 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2017, pp. 2921–2926.

[17] X. Bao, C. Su, Y. Xiong, W. Huang, and Y. Hu, "Flchain: A blockchain for auditable federated learning with trust and incentive," in *2019 5th International Conference on Big Data Computing and Communications (BIGCOM)*, 2019, pp. 151–159.

[18] Y. Li, C. Chen, N. Liu, H. Huang, Z. Zheng, and Q. Yan, "A blockchain-based decentralized federated learning framework with committee consensus," *IEEE Network*, vol. 35, no. 1, p. 234–241, Jan. 2021. [Online]. Available: http://dx.doi.org/10.1109/MNET.011.2000263

[19] H. Kim, J. Park, M. Bennis, and S.-L. Kim, "Blockchained on-device federated learning," 2019. [Online]. Available: https://arxiv.org/abs/1808.03949

[20] Q. Zheng, Y. Li, P. Chen, and X. Dong, "An innovative ipfs-based storage model for blockchain," in *2018 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*, 2018, pp. 704–708.

[21] J. Benet, "Ipfs - content addressed, versioned, p2p file system," 2014. [Online]. Available: https://arxiv.org/abs/1407.3561

[22] X. Wu, Z. Wang, J. Zhao, Y. Zhang, and Y. Wu, "Fedbc: Blockchain-based decentralized federated learning," in *2020 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA)*, 2020, pp. 217–221.

[23] H. Zhang, S. Jiang, and S. Xuan, "Decentralized federated learning based on blockchain: concepts, framework, and challenges," *Computer Communications*, vol. 216, pp. 140–150, 2024. [Online]. Available: https://doi.org/10.1016/j.comcom.2023.12.042

[24] I. Hegedüs, G. Danner, and M. Jelasity, "Gossip learning as a decentralized alternative to federated learning," in *IFIP International Conference on Distributed Applications and Interoperable Systems*, 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:174800884

[25] I. Hegedűs, G. Danner, and M. Jelasity, "Decentralized learning works: An empirical comparison of gossip learning and federated learning," *Journal of Parallel and Distributed Computing*, vol. 148, pp. 109–124, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0743731520303890

[26] C. Thapa, M. A. P. Chamikara, S. Camtepe, and L. Sun, "Splitfed: When federated learning meets split learning," 2022. [Online]. Available: https://arxiv.org/abs/2004.12088

[27] P. Vepakomma, O. Gupta, T. Swedish, and R. Raskar, "Split learning for health: Distributed deep learning without sharing raw patient data," 2018. [Online]. Available: https://arxiv.org/abs/1812.00564

[28] C. Hu, J. Jiang, and Z. Wang, "Decentralized federated learning: A segmented gossip approach," 2019. [Online]. Available: https://arxiv.org/abs/1908.07782

[29] Z. Tang, S. Shi, B. Li, and X. Chu, "Gossipfl: A decentralized federated learning framework with sparsified and adaptive communication," *IEEE Transactions on Parallel and Distributed Systems*, vol. 34, no. 3, pp. 909–922, 2023.

[30] A. Janjua, S. Dhalla, S. Gupta, and S. Singh, "A blockchain-enabled decentralized gossip federated learning framework," in *2023 International Conference on Networking and Communications (ICNWC)*, 2023, pp. 1–7.

[31] A. R. Elkordy, Y. H. Ezzeldin, S. Han, S. Sharma, C. He, S. Mehrotra, and S. Avestimehr, "Federated analytics: A survey," 2023. [Online]. Available: https://arxiv.org/abs/2302.01326

[32] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," in *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 2007, pp. 1027–1035.

[33] Wikipedia contributors, "Merkle Tree — Wikipedia, The Free Encyclopedia," https://en.wikipedia.org/wiki/Merkle_tree#, [Online; accessed 25-May-2025].

[34] Z. Cao and L. Liu, "The paillier's cryptosystem and some variants revisited," 2015. [Online]. Available: https://arxiv.org/abs/1511.05787

[35] D. Cajaraville-Aboy, A. Fernández-Vilas, R. P. Díaz-Redondo, and M. Fernández-Veiga, "Byzantine-robust aggregation for securing decentralized federated learning," 2024. [Online]. Available: https://arxiv.org/abs/2409.17754

[36] B. Osting and T. H. Reeb, "Consistency of dirichlet partitions," 2017. [Online]. Available: https://arxiv.org/abs/1708.05472

[37] Q. Li, Y. Diao, Q. Chen, and B. He, "Federated learning on non-iid data silos: An experimental study," 2021. [Online]. Available: https://arxiv.org/abs/2102.02079