The ISC Int'l Journal of Information Security
ISeCure

Manuscript template of ISeCure Journal (pp. 1–15)
http://www.isecure-journal.org

HashLearner: A Secure Decentralized Learning Framework Based on HashGraph

Keyhan Mohammadi [1], and Ehsan Kozegar [2] Reza Ebrahimi Atani [3,*]
[1] *Department of Computer Engineering, Faculty of Engineering, University of Guilan, Rasht, Iran*
[2] *Department of Computer Engineering, Faculty of Technology and Engineering-East of Guilan, University of Guilan, Rudsar, Guilan, Iran*
[3] *Department of Computer Engineering, Faculty of Engineering, University of Guilan, Rasht, Iran*

ARTICLE INFO.

Keywords:
Federated Learning, HashGraph, Decentralized Learning, Security, Privacy, Machine Learning.

Abstract

Federated learning enables collaborative model training without centralized data collection, but existing frameworks rely on a central server, introducing risks of single points of failure, adversarial manipulation, and privacy leakage. To address these challenges, we propose HashLearner, a secure decentralized learning framework that utilizes the HashGraph consensus protocol for model aggregation without trusted authorities. HashLearner introduces two key innovations: (i) a consensus-driven decentralized aggregation mechanism resilient to Byzantine adversaries, and (ii) a privacy-preserving shuffling strategy that mitigates gradient reconstruction and poisoning attacks. To handle heterogeneous data distributions, the framework further employs transfer learning–based personalization. The simulation results of HashLearner, tested on benchmark Kaggle datasets, demonstrate that the platform maintains high accuracy while significantly enhancing scalability, security, and privacy. These findings indicate that HashLearner provides a practical path toward scalable, privacy-preserving, and trustworthy decentralized federated learning.

1 Introduction

The deep learning community increasingly seeks decentralized solutions that allow organizations to jointly train models and perform computations while operating in untrusted environments. Some approaches to solve this problem have been proposed in recent years, including Federated Learning (FL) [1], [2], [3], [4] and Gossip Learning (GL) [5], [6], [7]. The increasing demand for privacy-preserving machine learning has accelerated the rise of FL, where multiple organizations collaboratively train a shared model without exchanging raw data. By keeping sensitive datasets local, FL mitigates risks associated with centralized data collection, such as unauthorized access, privacy breaches, and regulatory non-compliance. These approaches aim to distribute the training phase across a decentralized, multi-organizational network of nodes, where each node can have a different owner. In recent years, FL has emerged as a promising paradigm for collaborative machine learning, enabling multiple participants to train a shared model without exposing their private data. By decentralizing the training process, FL addresses critical privacy concerns and reduces the risks associated with data centralization. Despite these advantages, most FL frameworks rely on a central aggregation server to coordinate training. This dependence introduces critical

* Corresponding author.
Email addresses: keyhanmohammadi@webmail.guilan.ac.ir, Kozegar@guilan.ac.ir, rebrahimi@guilan.ac.ir
ISSN: 2008-2045

limitations:

- Single point of failure: If the server is compromised or fails, the system collapses.
- Adversarial vulnerability: Malicious participants can launch poisoning or reconstruction attacks, such as those using generative adversarial networks (GANs).
- Privacy risks: The server may infer sensitive information from model updates.
- Scalability challenges: Centralized coordination struggles to handle large numbers of clients in real-world distributed networks.

While alternatives such as gossip learning and blockchain-based FL have been proposed, gossip learning suffers from slow convergence and inconsistency under heterogeneous data. In contrast, blockchain-based solutions face scalability bottlenecks due to latency and resource-intensive consensus mechanisms.

This paper proposes a Secure Decentralized Learning Framework Based on HashGraph called HashLearner, which eliminates the dependency on a central server and leverages the unique properties of HashGraph, a distributed ledger technology known for its high throughput, fairness, and Byzantine fault tolerance. By integrating HashGraph into the FL process, the proposed framework ensures secure and efficient coordination among participants, even in the presence of adversarial actors. The decentralized nature of HashGraph not only enhances the resilience of the system but also mitigates the risks associated with centralized control, such as data manipulation and unauthorized access. A key contribution of this work is the framework's ability to counteract GAN attacks, which have become a significant threat in FL environments. GAN attacks exploit the collaborative nature of FL to generate malicious updates that can compromise the integrity of the global model.

The proposed framework incorporates advanced probabilistic consensus mechanisms to detect and neutralize such attacks, ensuring the reliability and trustworthiness of the learning process. The proposed framework offers several advantages over traditional FL systems, including improved scalability, enhanced security, and greater resistance to adversarial interference. By removing the central server and leveraging HashGraph's decentralized architecture, the proposed approach paves the way for more secure and privacy-preserving machine learning in distributed environments. This paper explores the design, implementation, and evaluation of the framework, demonstrating its effectiveness in mitigating GAN attacks and its potential to advance the field of FL.
The rest of the paper is structured as follows: Section 2 reviews related work in this area. Section 3 introduces some preliminary topics necessary for understanding our approach, and the proposed HashLearner platform is explained in detail. Section 4 focuses on security and performance analysis of the proposed scheme. Finally, Section 5 discusses the conclusions drawn from this research.

2 Related Works

Two prominent approaches in the decentralized learning domain are Gossip Learning and Federated Learning, each offering unique advantages and challenges in decentralized settings. Gossip Learning [6] is a fully decentralized approach where participants (nodes) exchange model updates directly with their neighbors in a peer-to-peer manner. This method eliminates the need for a central coordinator, making it highly scalable and resilient to single points of failure. Gossip protocols rely on randomized communication to propagate updates across the network, ensuring that knowledge is disseminated efficiently. Gossip learning faces two major challenges: (i) poor performance on heterogeneous datasets, where local data distributions vary significantly across nodes, and (ii) slower convergence compared to centralized or semi-decentralized FL approaches. In gossip learning, each node trains its local model using its local dataset, similar to FL, but there is no global initial model, and each node initializes the global model's initial weights independently. After training local models, each node exchanges its model's weights with other nodes over multiple rounds, and at each round, it merges its model with the other node's model. After several rounds, the models gradually converge toward a similar distribution across multiple nodes. Hence, GL may not perform well in heterogeneous networks with varying computational resources and communication delays because of the lack of a structured coordination mechanism and inconsistencies in model convergence [7].
On the other hand, FL has emerged as a widely adopted framework for decentralized learning, particularly in scenarios where data privacy is paramount. In FL, the objective is to train a high-quality centralized shared model while training data is distributed across a large number of clients with unreliable and relatively slow network connections, under the coordination of a central server. The server aggregates local model updates from participants and distributes the global model back to them.
While FL has demonstrated success in applications such as mobile devices and healthcare, its reliance on a central server introduces several limitations. These include vulnerability to single points of failure, potential privacy breaches, and susceptibility to poisoning and adversarial attacks, such as a reconstruction

attack. Poisoning attacks can target two different objectives. One is to corrupt the model by inserting erroneous data with an incorrect data distribution into the dataset, which is called a passive attack, and the other is to hide a backdoor inside the trained model by employing specific labels or data features in the dataset, which is called an active attack. In the reconstruction attack, the adversary attempts to reconstruct a shadow model based on the target model weights or gradients to extract information about the data in the dataset, such as dataset distribution, existence of specific data samples in the dataset, or certain properties of the dataset. For example, the adversary can determine whether an image exists in the dataset or identify the number of people with specific properties in the dataset. Reconstruction attacks can be implemented by employing GANs.
In GAN attacks, malicious participants generate fake updates to manipulate the global model, compromising its integrity and performance. A GAN comprises a generator and a discriminator trained in adversarial competition: the generator produces synthetic samples, while the discriminator attempts to distinguish them from real data. This dynamic enables adversaries in FL to craft poisoned updates or reconstruct private data. Conversely, the discriminator attempts to identify the fake samples generated by the generator. The competition between these two components leads to learning more features and advancement of the generator to create fake samples that are highly similar to real samples. In the reconstruction attack, the adversary can use GANs to create a shadow model and extract information about the dataset of a node, as mentioned previously. Multiple countermeasures have been proposed against these attacks, such as encryption, shuffling, differential privacy, and noise injection into the dataset, which have been helpful in making the system safer against some of the mentioned attacks [3]. To address these challenges, recent research has explored the integration of blockchain technology into FL, aiming to decentralize the aggregation process and enhance security. Blockchain-based FL systems replace the central server with a distributed ledger, enabling transparent and tamper-proof record-keeping of model updates. However, traditional blockchain systems often suffer from scalability issues, high latency, and energy inefficiency, which hinder their applicability in large-scale FL scenarios. In this context, HashGraph emerges as a promising alternative to blockchain for decentralized learning [8]. HashGraph [9] is a distributed ledger technology that utilizes a gossip-about-gossip protocol and virtual voting to achieve consensus in a highly efficient and secure manner. Unlike blockchain, HashGraph does not rely on resource-intensive proof-of-work mechanisms, making it more scalable and energy-efficient. Its inherent properties, such as Byzantine fault tolerance, fairness, and asynchronous communication, make it suitable for decentralized learning frameworks.
HashGraph offers higher throughput than proof-of-stake blockchains and is therefore well suited for AI training networks in which multiple models are trained simultaneously. Moreover, it provides asynchronous Byzantine fault tolerance, which the HashGraph designers describe as the strongest attainable level of security in asynchronous consensus. HashGraph's higher throughput makes it particularly effective for implementing AI training platforms across multiple organizations that need to train or expand AI models rapidly, as they may need to transfer many models simultaneously in each training round.
HashGraph is a decentralized network architecture consisting of nodes, each holding a copy of the HashGraph data, including its events graph. HashGraph stores data in a secure and decentralized manner, making it difficult for attackers to corrupt the stability and consistency of the ledger, including the order of events.
This research aims to improve decentralized learning approaches for aggregating a global model without central coordination. For example, in the FedHealth paper [5], the proposed approach requires a central server to aggregate the global model at the end of the training phase. As shown in Figure 1, FedHealth proposed a federated learning framework enhanced by transfer learning, optimized for healthcare model training and deep learning approaches. The authors employed transfer learning to adapt the global aggregated model to each node's local data distribution by training the aggregated model on local datasets separately at each node [5].

Related work includes blockchain-based federated learning (BCFL) [10], which explores similar decentralization concepts. In federated learning frameworks, the aggregator plays a crucial role by collecting and combining training results from local sources to update the global model. However, this central aggregator introduces reliability, security, and availability risks. A compromised aggregator can jeopardize the entire federated learning system, making it vulnerable to adversarial attacks. Potential threats include: Malicious aggregation behavior Network connectivity disruptions and denial-of-service attacks External security breaches and exploits [10].
To address these challenges, the authors implemented blockchain technology to finalize and share models through an immutable distributed ledger, an approach similar to HashLearner's use of HashGraph. However, HashGraph achieves higher throughput than

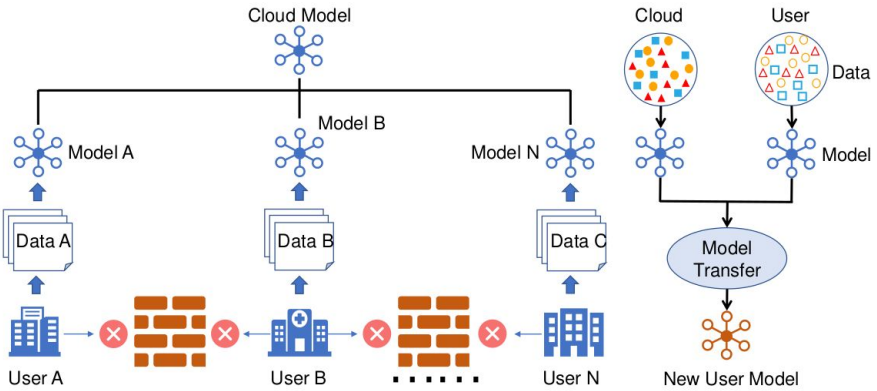

Figure 1. Overview of the FedHealth framework [5]

blockchain due to its asynchronous consensus architecture.

Another blockchain-based FL framework [8] secures IoT devices by utilizing them as blockchain nodes and employing a blockchain consensus mechanism for model sharing. While conceptually similar to previous work, HashLearner demonstrates superior speed owing to its consensus mechanism.

As noted previously, federated learning architectures are inherently vulnerable to both distributed denial-of-service (DDoS) attacks and privacy violations, including model data leakage. These vulnerabilities arise from malicious activities such as attempts to alter or steal confidential client data through manipulated model updates and weights. Furthermore, centralized FL systems face scalability challenges, particularly when handling the increasing volume of updates from growing IoT networks, which demand substantial computational resources [11], [12].

Blockchain technology, as a decentralized and immutable ledger system, offers several advantages, including decentralization, tamper resistance, and enhanced security. These properties effectively address the limitations of centralized FL systems that rely on a single server under a central authority. By eliminating dependence on a central server, the blockchain mitigates single points of failure and associated security risks.

The FeDis framework represents an approach that integrates federated learning with a distributed ledger to address security and trust concerns in collaborative machine learning. The framework trains a global model by iteratively aggregating local gradient models contributed by various participants. A notable feature of FeDis is its ability to handle heterogeneous data and devices, a common challenge in real-world federated systems. FeDis strengthens security and trust through the use of a distributed ledger. Participants' local model weights are encrypted before being saved to the ledger. The server then retrieves this encrypted data for aggregation to form the global model. This process adds a layer of traceability and security, thereby increasing user trust in the system [13].

DFL is a blockchain architecture designed to improve the efficiency of federated learning (FL). Instead of serving as a traditional distributed ledger, the blockchain functions as a distributed proof of contribution to the machine learning (ML) model. A key innovation is its asynchronous block generation capability, enabling each node to create blocks independently, which significantly enhances overall FL efficiency. The architecture's stability and robustness are addressed through a node reputation strategy and a weighted FedAvg implementation. These features are particularly effective in mitigating the challenges of non-I.I.D. (non-independent and identically distributed) datasets and defending against model poisoning attacks. From an ML perspective, the primary contribution is an asynchronous, gossip-based ML training method that uses the blockchain to store a verifiable proof of each node's contribution to the model [14].

Another study introduces a secure Federated Learning (FL) algorithm called MPCFL. MPCFL is built on the principles of secure multi-party computation (MPC) and secret sharing. The algorithm uses the Sharemind MPC framework to combine local model updates into a global model securely. MPCFL is designed to address common FL security issues, such as inference attacks, gradient leakage attacks, model poisoning, and model inversion [15].

The integration of blockchain and FL enables the development of decentralized architectures with enhanced security, improved privacy preservation, and greater reliability, which is particularly valuable in untrusted environments such as IoT networks that may contain adversarial nodes [8]. HashGraph delivers these same benefits while offering faster finalization and greater throughput.

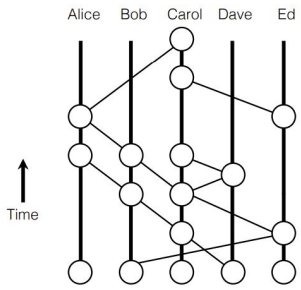

Figure 2. The HashGraph consensus schematic [9]

```
run two loops in parallel:
    loop
        sync all known events to a random member
    end loop
    loop
        receive a sync
        create a new event
        call divideRounds
        call decideFame
        call findOrder
    end loop
```

Figure 3. The HashGraph consensus algorithm [9]

The Swirlds HashGraph consensus mechanism provides a solution for replicated state machines, specifically designed to ensure Byzantine fault tolerance. Its fairness characteristic is particularly notable, as adversaries find it extremely difficult to manipulate transaction ordering during the consensus process. The system operates asynchronously without leaders, round-robin structures, or proof-of-work mechanisms, achieving eventual consensus with probability one while maintaining high performance in fault-free conditions.

At its core, the system utilizes a gossip protocol where participants do more than simply broadcast transactions - they also share information about the gossiping process itself. Through this process, participants collectively construct a HashGraph that records all gossip exchanges. This structure enables Byzantine agreement through a "virtual voting" mechanism within the HashGraph consensus.

In this system, Alice does not directly send ballots to Bob. Instead, Bob determines Alice's hypothetical ballot based on his knowledge of what Alice knows, gathered during the gossip process. This innovative approach achieves fair Byzantine agreement on a comprehensive transaction order while maintaining minimal communication overhead beyond the transactions themselves. Operating asynchronously, HashGraph establishes consensus without predefined paths. Its probabilistic nature virtually guarantees that Byzantine agreement will always be reached [9].

Figure 2 illustrates a HashGraph consensus example involving five nodes and the finalization of their events (blocks), while Figure 3 outlines the key components of the algorithms. The gossip history is represented as a directed graph in which each participant is mapped to a distinct vertical column of vertices (nodes). For example, if node Alice receives gossip from node Bob including all of Bob's knowledge, this exchange is modeled and stored as a vertex in Alice's column. This vertex then generates two downward-pointing edges, connecting it to the most recent prior gossip instances from both Alice and Bob.

HashLearner employs HashGraph as its consensus mechanism, eliminating the need for a central server (or cloud) in the federated learning process. A central server could act as an adversary, introducing vulnerabilities such as backdoor planting (poisoning) or corrupting model performance. To mitigate this, HashLearner introduces a validation step during aggregation, performed by multiple replicated multi-organizational nodes, similar to blockchain's voting mechanism.

The system is fault-tolerant, meaning it can withstand adversarial behavior as long as no more than half (50%) of the randomly chosen validators are malicious in any given aggregation step. If more than half of the validators are adversaries, the voting process fails.

To improve aggregation performance, especially for resource-constrained IoT nodes with limited computation power, we introduced a sharding step. Sharding reduces the need for each node to aggregate large numbers of models simultaneously, enhancing efficiency and benefiting low-memory nodes in the federated network. The primary weakness is that the system can fail if at least 33% of the nodes are adversarial. Although the aggregation process itself tolerates up to 50% malicious voters, the underlying HashGraph consensus is aBFT (asynchronous Byzantine Fault Tolerant) and can only resist up to 33% adversarial nodes. However, due to the asynchronous nature of HashGraph, controlling virtual voting or executing cheating attacks is inherently difficult. The key strength of this approach is its ability to remove reliance on an untrusted central server, enabling a masterless mesh of servers owned by different organizations to train AI models collaboratively without oversight from a central authority. A similar approach is now being adopted in AI training processes implemented on blockchains.

As mentioned in [5] and [16], FL is a technique to overcome the data isolation issue, making it possible to build models using data contributed by users across the network. Yet another key factor is the personalization of the applied solutions. Even if the cloud model can be used directly, it may not perform well for a

particular user. This is because the data distribution of individual users differs from that of the server's aggregated storage. The shared model trained on the server only captures the overall feature distribution across all users and therefore performs poorly in identifying the characteristics of specific users [5].

Transfer learning has been successfully applied in frameworks such as FedHealth, where trained models are adapted to local datasets to improve personalization and efficiency. HashLearner builds on this idea by using the same dataset and CNN model while eliminating the reliance on a central server. Alternative decentralized approaches have also been explored, most notably gossip learning. In gossip learning, nodes exchange model updates randomly with peers, enabling fully decentralized training without central coordination. While this design offers scalability and robustness, it suffers from slow convergence and the absence of a unique global model—each node retains its own final model, which only converges in distribution. Moreover, gossip learning cycles are asynchronous, and node selection relies on a sampling service [6, 17, 18]. HashLearner addresses these limitations through a decentralized aggregation mechanism that preserves scalability and robustness while ensuring faster convergence and producing a shared global model without requiring a central server.

In this paper, we build on these advancements by proposing HashLearner, a Secure Decentralized Learning Framework Based on HashGraph that eliminates the central server in federated learning and leverages HashGraph's consensus mechanism to coordinate model updates. By doing so, our framework addresses the limitations of traditional FL systems, such as vulnerability to GAN attacks and single points of failure, while maintaining the privacy and scalability benefits of decentralized learning. The integration of HashGraph ensures secure and efficient aggregation of model updates, even in adversarial environments, paving the way for a more robust and trustworthy federated learning paradigm.

3 Materials and Methods

In this section, we first introduce the dataset used in this paper and then describe the details of the base CNN classifier trained on each node. Afterwards, the main novelty of the presented work, named HashLearner, is described in detail.

3.1 Dataset Description

Similar to FedHealth, we use a dataset on Kaggle [19] referred to as "human activity recognition with smartphones." For this dataset, a study was conducted involving 30 participants aged 19 to 48. The subjects engaged in six different activities: walking on level ground, ascending stairs, descending stairs, sitting, standing, and lying down. Throughout these activities, they wore a Samsung Galaxy S II smartphone secured at their waist. The device's built-in accelerometer and gyroscope collected data on 3-axial linear acceleration and 3-axial angular velocity at a steady 50Hz frequency. To ensure accurate data labeling, all sessions were recorded on video. The dataset was then randomly split, with 70% allocated for training and the remaining 30% reserved for testing. The raw sensor data underwent initial processing to reduce noise. The signals were divided into fixed-width sliding windows of 2.56 seconds, with a 50% overlap between consecutive windows (resulting in 128 readings per window). A Butterworth low-pass filter separated the acceleration signal into its body motion and gravitational components. Because gravitational force is primarily associated with low-frequency signals, a filter with a 0.3 Hz cutoff was applied. From each window, a set of features was extracted by computing various time-domain and frequency-domain variables. For a fair comparison with FedHealth, we similarly extracted five topics from the mentioned dataset (i.e., subject IDs 26 to 30) and treated them as isolated customers (corporations or nodes) that cannot share records due to privacy constraints. Each node has approximately 400 training samples. It is worth noting that the dataset we used is raw and non-preprocessed, which makes training more challenging and demonstrates the robustness of the proposed approach compared to preprocessed visual datasets.

3.2 The proposed method

As mentioned in the Introduction, the main purpose of this study is to propose a novel approach for removing the cloud server in FedHealth to increase security without loss of performance. For fair comparisons with FedHealth, we use a shallow CNN classifier similar to the one used by FedHealth on each node to classify six categories for human activity recognition. The main contribution of this paper, HashLearner, is described in Section 4-2-2. HashLearner completely removes the cloud server used in FedHealth in a decentralized manner to prevent attacks on the server. The details of the aforementioned methods are explained in the following subsections. The remainder of this section presents the details required for implementing the HashLearner platform. Since this paper is based on an industrial software project, the source code of this work is also available at [20].

Table 1. Hyperparameters of the base CNN

Hyperparameter	Value
$kernels$	$(16, 32, 64)$
$Kernelsize$	3
$Poolsize$	2
$neuronsfordenselayers$	$(64, 32)$
$Optimizer$	SGD
$Learningrate$	0.01
$epochsforglobalmodel$	80
$Batchsize$	64

3.3 The Base CNN classifier

The deep learning model used in this paper is a Convolutional Neural Network (CNN), which is well suited for extracting local data relationships within the input. For fair comparisons, we adopted the same CNN classifier used in FedHealth, as shown in Figure 4. This architecture comprises convolution, pooling, and fully connected (dense) layers. After training, the system freezes the convolution and pooling layers and transfers the learned features of the local dataset to the global model more effectively by applying transfer learning to retrain only the fully connected layers at the end of the model. Hyperparameters of the base CNN classifier are summarized in Table 1.

3.4 HashLearner

The proposed approach in this research is named HashLearner, which employs HashGraph and secure decentralized random voting to create a safe and consistent way to share local models and the aggregated model in multi-step aggregation (the number of steps depends on the number of nodes, as the main purpose of multi-step aggregation is to distribute the computation cost as a trade-off with the security of the aggregation process). The training phase of the proposed HashLearner algorithm follows the steps in Figure 5. Each step in this diagram is described in detail below.

(1) Generate an initial model with random weights on each node separately as an initial global model.
(2) Each node submits its global model to HashGraph in a decentralized manner. Using HashGraph in this step helps the network to prevent adversary nodes from corrupting the sharing process.
(3) Conduct an election among nodes to generate a shared, consistent, randomly ordered array of node IDs.
(4) Each node combines the models on the HashGraph by integrating layers based on the elected order of nodes, which produces a single unique global model (containing collaboratively generated random weights) on all nodes. As a result, all nodes now share the same global model for training.
(5) Each node trains the global model on its local dataset, resulting in a local model. Then, it submits its local model to the HashGraph.
(6) After all local models exist on the HashGraph, nodes conduct another election. The result is another shared, randomly ordered array of node IDs. The system self-shards the node list based on each node's position in the produced order (e.g., if the node count is 100, there can be 4 shards of 25 grouped nodes, each shard as a subarray of the shared order: 0 to 24 for shard 1, 25 to 49 for shard 2, etc.). After sharding, in each shard the first 3 nodes in the shared order are designated as shard aggregators, which validate each other's work proof. Each shard aggregator collects local models of nodes in its shard and aggregates them simultaneously (in this research's implementation, averaging is employed for aggregation, but any other aggregation method can be used). Afterwards, all aggregators of all shards submit their produced shard models to the HashGraph.
(7) Nodes conduct another election to produce a shared, randomly ordered array of node IDs and designate the first 3 nodes in the array as global aggregators. Global aggregators retrieve all shard models from the HashGraph. They compare the shard models of each shard simultaneously and select the most voted one (the most repeated model in the list of shard models for each shard). They then aggregate the elected shard models (each shard has one elected shard model) and submit the aggregated global model to the HashGraph simultaneously.
(8) All nodes retrieve the aggregated global models and independently select the most repeated one (most voted) as the global model.
(9) Repeat steps 2 to 8 for any number of rounds for further convergence of the model. Notably, training on the local dataset in each round applies transfer learning, helping the global model adapt to the distribution of different local datasets.

Note: In this specific example, the sharding hyperparameters such as shard size, shard aggregator count, and global aggregator count are not optimized for security. The aggregator count in both shard and global aggregation must exceed 2 nodes; however, the

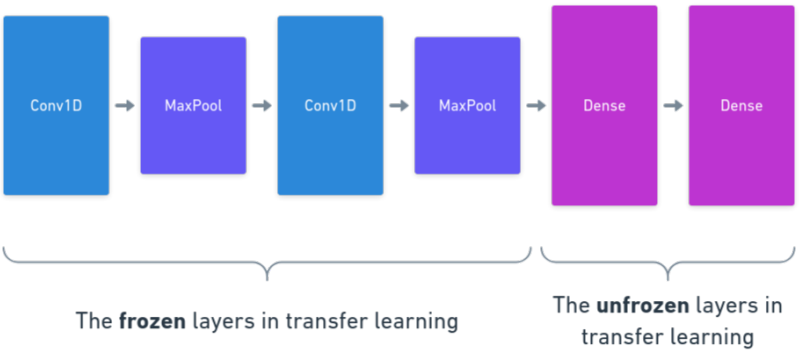

Figure 4. The base CNN classifier on each node

algorithm remains the same. Simply adjusting these parameters in the system settings is sufficient for optimization and increasing security. Therefore, no changes to the software code are needed to improve security, and additional nodes can be incorporated easily by modifying the configuration object containing the three mentioned hyperparameters.

This algorithm guarantees that all global models on all nodes are identical after the execution of these steps. The election process for producing a shared, randomly ordered array of node IDs uses the "commit & reveal scheme."

In the commit & reveal scheme:

(1) Nodes generate a random number and then hash it.
(2) They share the produced hash value.
(3) Then they share the original random number.
(4) Each node validates the election by comparing the random number with its hash for each peer. If all hashes match their corresponding random numbers, the election is validated.
(5) Each node performs an XOR operation on the group of random numbers to produce a single random number. The election guarantees that all nodes now hold the same number.
(6) Finally, each node scales the produced number from a range of the maximum number with the same bit count as the number of nodes to the actual number of nodes, resulting in an index in the range 0 to node_count − 1.
(7) Nodes then treat the node at this index in the node list as the elected node for this round and append its ID to the shared randomly ordered array of node IDs.
(8) Steps 1 to 7 are repeated node_count times to produce a shared random order of nodes as the election result for the various use cases in this system.

As mentioned in the Dataset section, we have conducted a specific setup to demonstrate the ability of the proposed method in which the node count was not high, so the shard size and aggregator count were the same. However, the primary functionality of the HashLearner network is in a high node count environment (e.g., 100 nodes with 4 or 5 shards of 25 or 20 nodes in each shard), resulting in distribution of aggregation across different shards on a network in which each node has limited resources (due to limited resources, they cannot aggregate all local models on their devices).

Assuming we have 5 nodes—Node A, Node B, Node C, Node D, and Node E—each with a subject as its local dataset, after the shard election they are mapped to indices 0 to 4 as Node 0 through 4. Table 2 shows how the sharding and aggregator selection occurred.

It is worth mentioning that GAN-based attacks aimed at reconstructing models can be prevented by shuffling peer layers of models between nodes in a peer-to-peer manner. After sharding is completed, each node randomly selects another node and exchanges a specific layer of its model with it. This ensures that while the models are shuffled, the aggregation result remains unchanged and the model's integrity is preserved. Additionally, poisoning the model via GAN-based attacks is ineffective because aggregation is validated through consensus, and the final result is determined by majority vote.

4 Results and Discussions

We used a well-known classification performance metric, accuracy, to compare FedHealth and HashLearner. Accuracy is the percentage of correct predictions made by a machine learning model out of the total number of predictions.

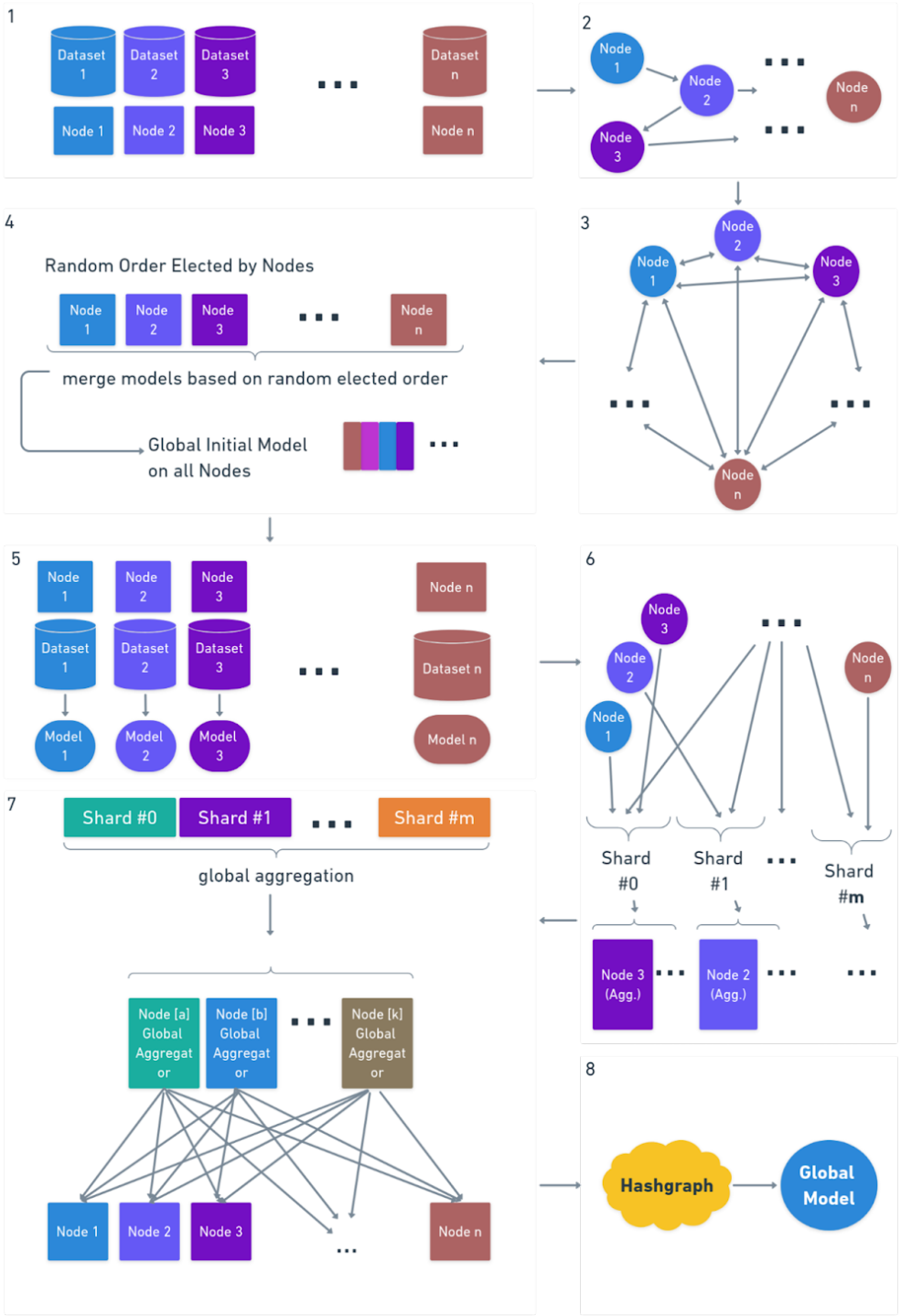

Figure 5. Overview of the proposed HashLearner approach

Table 2. The state of the system at each round after sharding for this 5 node example (in a higher node count not all nodes will be chosen as aggregator)

Subjects	Subject 0	Subject 1	Subject 2	Subject 3	Subject 4
Nodes	Node 0	Node 1	Node 2	Node 3	Node 4
Shard index	0	0	0	1	1
Is Aggregator	yes	yes	yes	yes	yes

Table 3. Hyperparameters of the proposed HashLearner algorithm

Hyperparameter	Value
Node Count	100
Global Agg. Count	5
Shard Size	25
Shard Count	4
Shard Agg. Count	5
Election method	commit and reveal

In this experiment, we ran the FedHealth and HashLearner methods 20 times each, separately. The accuracy of the final global model for FedHealth and HashLearner is 92.26% ± 0.98 and 93.57 ± 0.51, respectively. Notably, the accuracy of HashLearner remained nearly the same despite removing the FedHealth cloud server; the small numerical differences are due to randomness in calculations and model generation, indicating that model performance was preserved (neither improved nor degraded). The accuracy, precision, recall, and F1-score of each of the five subject nodes are shown in Table 4. As shown in the table, the mean accuracies across all subjects are very close, which demonstrates the convergence of the global model on each node.

In addition to the above experiment, we statistically analyzed the difference between the competing methods using a paired t-test. A precondition for using the paired t-test is that both random variables should follow a normal distribution. Therefore, we applied the Anderson-Darling test for the null hypothesis that the accuracies obtained for FedHealth and HashLearner come from normal distributions. After applying the Anderson-Darling test, the null hypothesis was not rejected at the 5% alpha level. Therefore, the precondition for applying the paired t-test is satisfied. The null hypothesis for the paired t-test states that the accuracies of HashLearner and FedHealth come from independent random samples from normal distributions with equal means and equal but unknown variances. After applying this test, the null hypothesis was rejected at the 5% alpha level with a p-value of 5.6e-6, indicating that the accuracy of our proposed HashLearner is statistically superior to that of FedHealth.

4.1 Performance Analysis of the Proposed Method

The experiments in our study were conducted on a Google Colab virtual machine using a T4 GPU with 12 GB of RAM. The convergence speed of the proposed approach is provably the same as that of typical federated learning methods (e.g., FedHealth) because the only modification from basic federated learning is the shuffling process. This shuffling does not alter the median (or average) aggregation result, as the same layers are exchanged peer-to-peer between nodes. Sharding merely distributes the computational process across nodes without affecting the final result. Mathematically, the median of 10 numbers can be calculated either directly or by splitting them into two shards of 5 numbers each, first computing the median of each shard and then taking the median of those two results. The outcome remains identical.

Furthermore, the HashGraph consensus and voting mechanism does not modify the calculation itself; it only secures the process. Thus, there is no difference between traditional federated learning aggregation and HashLearner's aggregation in terms of results. However, HashLearner eliminates the need for a central server (third-party moderation), making it more secure and suitable for decentralized training across multiple organizations or communities without relying on a central authority.

At each round of training, approximately 3–4 seconds were added for the election, gossiping, and multi-step aggregation phase in a simulated local environment.
Although in a practical environment model serialization can be a bottleneck for node communications and model transfer, multiple solutions can be employed to address this problem, such as:

- Parallelized serialization (as the target object is a list of matrices).
- Parallelized model transfer over the network

Table 4. The accuracy of final local model of different nodes

Subjects	Subject A	Subject B	Subject C	Subject D	Subject E
Nodes	Node A	Node B	Node C	Node D	Node E
Accuracy	93.5 ± 1%	94.5 ± 1%	93.5 ± 2 %	93 ± 2 %	94 ± 1 %
F1 score	0.894	0.909	0.939	0.888	0.920
precision	0.896	0.910	0.941	0.889	0.921
recall	0.892	0.908	0.937	0.887	0.919

using multiple connections and channels.

These two approaches can reduce model transfer time over the network and accelerate HashGraph model submission. It is worth noting that the time required to finalize blocks in a HashGraph increases logarithmically relative to the number of nodes participating in the learning process. Sharding increases both consensus speed and aggregation speed by decreasing the number of nodes participating in the consensus as well as the number of models being aggregated. Reducing the number of nodes in the consensus introduces a trade-off between speed and security: it decreases security to increase speed. Larger shards are slower to finalize blocks and aggregate models but are more secure, as more nodes participate in consensus and more validators can be chosen for aggregating models within the shard.

Scalability and throughput: As demonstrated in the Swirlds HashGraph paper, the HashGraph consensus achieves higher throughput than other multiparty computation or blockchain alternatives. Additionally, HashGraph enables peers to submit blocks concurrently, which increases scalability efficiently. The sharding algorithm proposed in our paper further distributes load and computation bottlenecks, improving scalability compared to other approaches mentioned in the related work. HashLearner achieves higher throughput than FeDis [13] and DFL [14] because, instead of blockchain, it relies on HashGraph with asynchronous gossiping and block finalization and uses sharding to distribute the computation load across the network simultaneously. In an ideal situation, HashLearner can average models in fewer than 10 seconds for a single round of averaging (within a shard or globally). In real-world situations, network latency may degrade this process; however, this degradation applies to all approaches, including blockchains, and there is always a gap between ideal and real-world statistics. Furthermore, HashLearner outperforms MPCFL [15] in throughput and provides higher security through its use of a distributed ledger and a higher level of verification.

4.2 Security and Privacy Analysis of the Proposed Method

Federated learning is designed to enable collaborative model training across distributed nodes without sharing raw data. However, the involvement of an honest-but-curious cloud server—a server that follows the protocol correctly but may attempt to infer sensitive information from the shared data—introduces significant security risks. As mentioned earlier, the work presented in [5] is vulnerable to three critical attack vectors: GAN attacks, reconstruction attacks, and poisoning attacks. HashLearner uses a decentralized consensus mechanism (HashGraph) to ensure secure and transparent model aggregation and reduces the risk of single-point failures and attacks targeting a central authority. In this section, a formal security analysis of the HashLearner platform is presented:

4.2.1 Formal Security Model and Analysis

In this part of the paper, we formalize the security guarantees of the proposed method under standard cryptographic frameworks. We consider a decentralized federated learning system consisting of a set of nodes $\mathcal{N} = \{P_1, P_2, \ldots, P_n\}$, where each node P_i holds a private dataset D_i. The goal is to collaboratively train a global model M without exposing raw data. Each training round proceeds as follows:

(1) Each node trains a local model M_i on D_i.
(2) Updates are obfuscated using *peer-to-peer shuffling*.
(3) Nodes broadcast updates through the HashGraph gossip-about-gossip protocol.
(4) HashGraph consensus determines the final ordering of events.
(5) Aggregation produces the updated global model.

We assume a probabilistic polynomial time (PPT) adversary $\mathcal{A}$ controlling up to $t < n/3$ nodes, consistent with the fault tolerance of HashGraph consensus. The adversary may attempt:

- **Reconstruction attacks:** Inferring private data from shared updates (e.g., gradient inver-

sion, GAN-based attacks).

- **Poisoning/backdoor attacks:** Injecting manipulated updates to degrade accuracy or embed targeted misclassifications.
- **Message manipulation:** Delaying, reordering, or dropping messages to bias consensus.
- **Honest-but-curious behavior:** Following the protocol while attempting to extract information from others' updates.

The adversary is computationally bounded, and communication channels are authenticated but not confidential. We define the ideal functionality $\mathcal{F}_{HL}$ for HashLearner:

- **Inputs:** Each party P_i provides its local update M_i.
- **Process:**
  (1) $\mathcal{F}_{HL}$ applies randomized shuffling to anonymize updates.
  (2) An incorruptible consensus oracle determines the order of updates.
  (3) Secure aggregation produces the global model M.
- **Outputs:** The global model M is delivered to all parties.

In the ideal world, adversaries see only their own inputs and the final global model. But real-world execution of HashLearner:

- Shuffling is implemented through peer-to-peer exchanges.
- Consensus is achieved via HashGraph's gossip-about-gossip with virtual voting.
- Updates are broadcast and aggregated in the agreed-upon order.

In this setting, adversaries may observe update flows and corrupted nodes' contributions.

4.2.2 Security Argument Based on Ideal–Real Simulation

Assuming authenticated channels and the Byzantine resilience of HashGraph consensus, HashLearner securely realizes $\mathcal{F}_{HL}$ against any PPT adversary controlling fewer than $n/3$ nodes. This holds because:

(1) **Consensus integrity:** HashGraph achieves asynchronous Byzantine fault tolerance; thus, adversaries cannot bias the ordering of honest updates except with negligible probability.
(2) **Confidentiality of updates and reconstruction attacks:** Randomized shuffling ensures that adversaries cannot link specific updates to honest nodes or infer private data except with negligible probability.
(3) **Simulatability:** Any adversary's view in the real world (messages, corrupted updates, final model) can be simulated in the ideal world using only the global model and corrupted inputs. Hence, the distributions are indistinguishable.
(4) **Poisoning resistance:** Because aggregation requires consensus, adversarial influence is bounded, preventing domination of the global model. This also ensures that no single node can manipulate the aggregation result.
(5) **Resource efficiency balanced with security:** Nodes with limited resources, such as IoT nodes, can participate without overburdening a central server, as the aggregation workload is securely distributed.
(6) **Peer-to-peer communication:** Reduced latency and improved scalability compared to centralized systems. Removing the central server eliminates the central communication and data transfer proxy that could be insecure and vulnerable to data manipulation.
(7) **Transparent and auditable:** HashGraph's consensus mechanism ensures that all transactions, such as model updates, are recorded and verifiable, reducing the risk of malicious behavior. HashLearner can also serve as an intrusion detection system (IDS) for identifying malicious nodes in poisoning attacks.
(8) **Mitigates GAN attacks**: By distributing the aggregation process and obfuscating local updates, HashLearner reduces the risk of data leakage.
(9) **HashGraph consensus security:** HashGraph is fault-tolerant against up to 33% adversarial nodes, as it is an *asynchronous Byzantine Fault-Tolerant (aBFT)* protocol—the highest security level for asynchronous consensus. This security comes with the added benefit of high throughput, enabling HashGraph to finalize multiple blocks simultaneously while outperforming synchronous blockchains in speed.

Therefore, HashLearner is secure in the ideal–real paradigm.

4.2.3 Privacy Leakage Analysis

Although complete privacy cannot be guaranteed in gradient-based learning, we analyze the reduction in leakage through empirical evaluation.

- **Baseline FL (FedAvg):** It is vulnerable to membership inference attacks and has the lowest privacy in this comparison. Its core principle is that clients send only their model updates, not their raw data, to a central server. However, these updates, which are essentially averaged gradients, contain information about the data

used to calculate them. Attackers can exploit this by using various techniques to reverse-engineer the updates and infer private information [21].

- **Blockchain-based FL:** The use of immutable ledgers provides a high level of privacy defense; however, because no native obfuscation is applied, model weights recorded on the ledger remain exposed to potential inference attacks. In many blockchain-based FL designs, the encrypted or unencrypted model updates (gradients) are stored on a public ledger. Even if a third party cannot manipulate the data, they can access it. An adversary can analyze these publicly available gradient updates over time to perform gradient inversion attacks or membership inference attacks, similar to those in traditional FedAvg. Moreover, a blockchain-based system cannot inherently prevent participants from colluding. If a malicious client colludes with a miner node on the blockchain, they can still leak sensitive information. For example, a group of clients can aggregate their gradients and share them with an outside party, or a malicious client can submit malicious updates to reconstruct another client's data [22].
- **HashLearner:** The privacy of the shuffling system approaches random guessing, due to peer-to-peer shuffling of model weights and consensus-driven aggregation. This shuffling process conceals the identity of individual local models. Although the weights are shuffled, the final result is the same as if they had not been, because a non-weighted average is used to combine the models, which does not alter the final outcome. This is because only corresponding layers (e.g., the third layer of one model with the third layer of another) are shuffled with each other. The shuffling makes it impossible to link specific data to a particular model. Notably, a higher number of nodes and a higher number of layers in the deep learning model increase shuffling security and make it harder for an adversary to identify the actual models and infer data. Privacy is inherently preserved by the decentralized nature of HashGraph and the multi-step aggregation process based on a probabilistic mechanism; hence, no reliance on a trusted third party is required.

Thus, HashLearner significantly reduces the adversary's advantage while maintaining model utility. This framework offers significant advantages over traditional cloud-based FL systems in terms of security, privacy, scalability, throughput, transparency, and validation. It also eliminates risks associated with single points of failure and adversarial behavior. These properties make HashLearner a highly secure and scalable solution for federated learning. In contrast, cloud-based federated learning systems remain vulnerable to attacks targeting the central server and require additional privacy-preserving measures to mitigate risks. HashLearner's decentralized approach represents a paradigm shift in federated learning, addressing the limitations of traditional systems and paving the way for more secure and privacy-preserving collaborative learning frameworks.

5 Conclusion

In this work, we introduced HashLearner, a secure and decentralized federated learning framework that eliminates the reliance on a central server by leveraging the HashGraph consensus protocol. The framework integrates three key components: (i) decentralized aggregation via Byzantine fault-tolerant consensus, (ii) a peer-to-peer shuffling mechanism that obfuscates local model updates and mitigates reconstruction attacks, and (iii) a transfer learning–based personalization method to adapt the aggregated global model to heterogeneous client datasets. Our evaluation demonstrates that HashLearner maintains accuracy comparable to conventional FL while significantly improving robustness against poisoning and inference attacks, reducing privacy leakage, and scaling efficiently in distributed environments. These results indicate that HashLearner provides a practical pathway toward trustworthy and scalable federated learning in adversarial and resource-constrained settings.

References

[1] Jakub Konečný, H. Brendan McMahan, Felix X. Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency, 2017. URL `https://arxiv.org/abs/1610.05492`.

[2] Li Li, Yuxi Fan, Mike Tse, and Kuo-Yi Lin. A review of applications in federated learning. *Computers Industrial Engineering*, 149:106854, 2020. ISSN 0360-8352. doi: https://doi.org/10.1016/j.cie.2020.106854. URL `https://www.sciencedirect.com/science/article/pii/S0360835220305532`.

[3] Shui Yu and Lei Cui. *Security and Privacy in Federated Learning*. Springer, 2023. doi: 10.1007/978-981-19-8692-5.

[4] Keyhan Mohammadi and Reza Ebrahimi Atani. Sigma: A secure federated network gaming platform. In *2024 15th International Conference on Information and Knowledge Technology (IKT)*, pages 222–227, 2024. doi: 10.1109/IKT65497.2024.10892800.

[5] Yiqiang Chen, Xin Qin, Jindong Wang, Chaohui Yu, and Wen Gao. Fedhealth: A federated trans-

fer learning framework for wearable healthcare. *IEEE Intelligent Systems*, 35(4):83–93, 2020. doi: 10.1109/MIS.2020.2988604.

[6] Márk Jelasity, Spyros Voulgaris, Rachid Guerraoui, Anne-Marie Kermarrec, and Maarten van Steen. Gossip-based peer sampling. *ACM Trans. Comput. Syst.*, 25(3):8–es, August 2007. ISSN 0734-2071. doi: 10.1145/1275517.1275520. URL https://doi.org/10.1145/1275517.1275520.

[7] Alexandre Pham, Maria Potop-Butucaru, Sébastien Tixeuil, and Serge Fdida. Data poisoning attacks in gossip learning, 2024. URL https://arxiv.org/abs/2403.06583.

[8] Wael Issa, Nour Moustafa, Benjamin Turnbull, Nasrin Sohrabi, and Zahir Tari. Blockchain-based federated learning for securing internet of things: A comprehensive survey. *ACM Comput. Surv.*, 55(9), January 2023. ISSN 0360-0300. doi: 10.1145/3560816. URL https://doi.org/10.1145/3560816.

[9] Leemon Baird. The swirlds hashgraph consensus algorithm: Fair, fast, byzantine fault tolerance. *Swirlds Tech Reports SWIRLDS-TR-2016-01, Tech. Rep*, 34:9–11, 2016.

[10] Zhilin Wang, Qin Hu, Minghui Xu, Yan Zhuang, Yawei Wang, and Xiuzhen Cheng. A systematic survey of blockchained federated learning, 2024. URL https://arxiv.org/abs/2110.02182.

[11] Dun Li, Dezhi Han, Tien-Hsiung Weng, Zibin Zheng, Hongzhi Li, Han Liu, Arcangelo Castiglione, and Kuan-Ching Li. Blockchain for federated learning toward secure distributed machine learning systems: a systemic survey. *Soft Computing*, 26(9):4423–4440, 2022. doi: https://doi.org/10.1007/s00500-021-06496-5.

[12] Dinh C. Nguyen, Ming Ding, Quoc-Viet Pham, Pubudu N. Pathirana, Long Bao Le, Aruna Seneviratne, Jun Li, Dusit Niyato, and H. Vincent Poor. Federated learning meets blockchain in edge computing: Opportunities and challenges, 2021. URL https://arxiv.org/abs/2104.01776.

[13] Rafael Barbarroxa, João Silva, Luis Gomes, Fernando Lezama, Bruno Ribeiro, and Zita Vale. Fedis: Federated learning framework supported by distributed ledger. In *Blockchain and Applications, 5th International Congress*, pages 32–41. Springer Nature Switzerland, 2023. doi: 10.1007/978-3-031-45155-3_4.

[14] Yongding Tian, Zhuoran Guo, Jiaxuan Zhang, and Zaid Al-Ars. Dfl: High-performance blockchain-based federated learning. *Distrib. Ledger Technol.*, 2(3), September 2023. doi: 10.1145/3600225. URL https://doi.org/10.1145/3600225.

[15] Hiroki Kaminaga, Feras M. Awaysheh, Sadi Alawadi, and Liina Kamm. Mpcfl: Towards multi-party computation for secure federated learning aggregation. In *Proceedings of the IEEE/ACM 16th International Conference on Utility and Cloud Computing*, UCC '23, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400702341. doi: 10.1145/3603166.3632144. URL https://doi.org/10.1145/3603166.3632144.

[16] Saba Ameri and Reza Ebrahimi Atani. A novel decentralized privacy preserving federated learning model for healthcare application. In *2024 15th International Conference on Information and Knowledge Technology (IKT)*, pages 115–120, 2024. doi: 10.1109/IKT65497.2024.10892736.

[17] István Hegedűs, Gábor Danner, and Márk Jelasity. Gossip learning as a decentralized alternative to federated learning. In José Pereira and Laura Ricci, editors, *Distributed Applications and Interoperable Systems*, pages 74–90, Cham, 2019. Springer International Publishing. ISBN 978-3-030-22496-7. doi: 10.1007/978-3-030-22496-7_5.

[18] Roberto Roverso, Jim Dowling, and Mark Jelasity. Through the wormhole: Low cost, fresh peer sampling for the internet. In *IEEE P2P 2013 Proceedings*, pages 1–10, 2013. doi: 10.1109/P2P.2013.6688707.

[19] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra, and Jorge L. Reyes-Ortiz. Human activity recognition on smartphones using a multiclass hardware-friendly support vector machine. In José Bravo, Ramón Hervás, and Marcela Rodríguez, editors, *Ambient Assisted Living and Home Care*, pages 216–223, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. ISBN 978-3-642-35395-6. doi: 10.1007/978-3-642-35395-6_30.

[20] Keyhan Mohammadi, Reza Ebrahimi Atani. HashLearner, 2025. URL https://github.com/cosmopole-org/hash-learner. Accessed on 2025-04-01.

[21] Ligeng Zhu, Zhijian Liu, and Song Han. Deep leakage from gradients, 2019. URL https://arxiv.org/abs/1906.08935.

[22] Privacy-preserving in blockchain-based federated learning systems. *Computer Communications*, 222:38–67, 2024. ISSN 0140-3664. doi: https://doi.org/10.1016/j.comcom.2024.04.024.

Keyhan Mohammadi is a Ph.D. candidate in Computer Engineering at the University of Guilan, Rasht, Iran, since 2024. His research interests lie at the intersection of dis-

tributed systems and artificial intelligence, with a particular focus on decentralized AI. He works on topics including blockchain technologies, AI-driven distributed systems, and the design of secure and trustworthy autonomous AI-based ledgers.

Ehsan Kozegar was born in Ramsar, Iran, in 1986. He received the B.Sc. degree in computer engineering, and the M.Sc. and Ph.D. degrees in artificial intelligence from the Iran University of Science and Technology (IUST), Tehran, Iran, in 2009, 2011, and 2018, respectively. Since 2018, he has been a Faculty Member with the University of Guilan, Guilan, Iran. He is currently an Assistant Professor with the Faculty of Technology and Engineering, University of Guilan. His main research interests include machine learning, image processing, and medical image analysis.

Reza Ebrahimi Atani received his B.Sc. degree in Electronics Engineering from the University of Guilan in 2002. He earned his M.Sc. (2004) and Ph.D. (2010) degrees in Electronics Engineering from Iran University of Science and Technology (IUST), where his doctoral research focused on design and secure implementation of symmetric key cryptographic algorithms. During his Ph.D., he conducted research fellowships at FHNW (Switzerland) and KU Leuven (Belgium), specializing in design and secure implementations of stream ciphers. Currently an Associate Professor in the Computer Engineering Department at the University of Guilan, Dr. Atani leads the Computer Security Incident Response Teams (CSIRT) Research Laboratory. His research expertise spans: Design and cryptanalysis of security protocols for wired/wireless networks, AI-driven cybersecurity solutions and threat intelligence, Secure architectures for IoT ecosystems.