Expert Systems With Applications 238 (2024) 122210

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Privacy-preserving and Byzantine-robust Federated Learning Framework using Permissioned Blockchain

Harsh Kasyap, Somanath Tripathy [*]

Department of Computer Science and Engineering, Indian Institute of Technology Patna, India

ARTICLE INFO

Keywords:
Federated learning
Poisoning attack
Robustness
Inference attack
Privacy
Permissioned blockchain

ABSTRACT

Data is readily available with the growing number of smart and IoT devices. However, application-specific data is available in small chunks and distributed across demographics. Also, sharing data online brings serious concerns and poses various security and privacy threats. To solve these issues, federated learning (FL) has emerged as a promising secure and collaborative learning solution. FL brings the machine learning model to the data owners, trains locally, and then sends the trained model to the central curator for final aggregation. However, FL is prone to poisoning and inference attacks in the presence of malicious participants and curious servers. Different Byzantine-robust aggregation schemes exist to mitigate poisoning attacks, but they require raw access to the model updates. Thus, it exposes the submitted updates to inference attacks. This work proposes a Byzantine-Robust and Inference-Resistant Federated Learning Framework using Permissioned Blockchain, called PrivateFL. PrivateFL replaces the central curator with the Hyperledger Fabric network. Further, we propose *VPSA* (Vertically Partitioned Secure Aggregation), tailored to PrivateFL framework, which performs robust and secure aggregation. Theoretical analysis proves that *VPSA* resists inference attacks, even if $n-1$ peers are compromised. A secure prediction mechanism to securely query a global model is also proposed for PrivateFL framework. Experimental evaluation shows that PrivateFL performs better than the traditional (centralized) learning systems, while being resistant to poisoning and inference attacks.

1. Introduction

In the era of technological advancements, every individual owns smart devices and can generate data. These devices contribute to designing a better and more intelligent solution. However, government regulations and compliance like GDPR (European Commission, 2016) restrict online data sharing. Federated learning (FL) is a collaborative learning paradigm that keeps the data in place and training locally (McMahan, Moore, Ramage, & y Arcas, 2016). FL has been adopted across different fields, such as healthcare (Kulkarni, Kasyap, & Tripathy, 2021), malware detection (Singh, Kasyap, & Tripathy, 2020), transportation (Kong, Gao, Shen, Duan, & Das, 2021), fifth-generation (5G) wireless communication technology (Wang, Hu, et al., 2022).

Though FL is said to be privacy-preserving in nature, it suffers from different attacks, such as *poisoning* (Bagdasaryan, Veit, Hua, Estrin, & Shmatikov, 2020; Cao, Chang, Lin, Liu, & Sun, 2019; Fang, Cao, Jia, & Gong, 2020; Kasyap & Tripathy, 2022, 2024), *inference* (Luo, Wu, Xiao, & Ooi, 2021; Manna, Kasyap, & Tripathy, 2022; Shokri, Stronati, Song, & Shmatikov, 2017), *inversion and reconstruction* attacks (Fredrikson, Jha, & Ristenpart, 2015). FL suffers from poisoning attacks in the presence of malicious attackers, which degrades the model performance. Authors in Blanchard, El Mhamdi, Guerraoui, and Stainer (2017), Cao, Fang, Liu, and Gong (2021), Manna, Kasyap, and Tripathy (2021) and Yin, Chen, Kannan, and Bartlett (2018) proposed Byzantine-robust defenses to mitigate such attacks. However, these schemes investigate individual model updates, which exposes the local model updates to inference attacks. It has been observed that adversaries can gain insights and infer sensitive information from the participant's local model updates (Manna et al., 2022). Thus, it requires a secure aggregation strategy to ensure privacy for participants. Authors in Hao et al. (2019) and Wei et al. (2020) proposed mitigation techniques for inference attacks, but have no poisoning defense mechanism. Robust and secure aggregation is therefore essential to FL ensuring that the client's local model parameters are not compromised or leaked to curious clients or server. Additionally, it needs to detect and prevent poisoning attacks. By aggregating the model updates from multiple (untrusted) sources, these schemes should help in reducing the impact of noisy or biased updates, improving the model's overall performance.

* Corresponding author.
E-mail addresses: harsh_1921cs01@iitp.ac.in (H. Kasyap), som@iitp.ac.in (S. Tripathy).

1 https://hyperledger-Fabric.readthedocs.io/en/release-2.2/whatis.html

https://doi.org/10.1016/j.eswa.2023.122210
Received 15 July 2023; Received in revised form 24 September 2023; Accepted 16 October 2023
Available online 18 October 2023
0957-4174/© 2023 Elsevier Ltd. All rights reserved.

Further, the central server in traditional FL is a common failure point and could be malicious (lazy/biased) or curious. A malicious server targets to corrupt the integrity of the aggregated result. So, a fair aggregation strategy is required to establish a decentralized trust in the system. Authors in Lu, Huang, Dai, Maharjan, and Zhang (2019), Nguyen et al. (2021) and Wang et al. (2023), discussed how blockchain can be integrated with FL. Blockchain-based secure data aggregation strategy has been proposed for edge computing empowered IoT (Wang, Garg, et al., 2022). Meanwhile, the existing works (Kasyap & Tripathy, 2021; Weng et al., 2019) to integrate blockchain with FL, provide privacy and robustness, incurring significant computation and communication overheads. This is because integrating cryptographic protocols like, homomorphic encryption and secure multi-party computation, results in increased processing time and inflated data transmission. Besides this, storage, latency and scalability have also been a concern for integrating FL and blockchain (Kasyap, Manna, & Tripathy, 2022). Further, blockchain is primarily classified as public (*e.g.* Ethereum Wood et al., 2014) and permissioned (*e.g.* Hyperledger Fabric Androulaki et al., 2018) blockchain. Ethereum introduced smart contracts, which allow the writing of business logic. Hyperledger Fabric also introduces a similar contract called chaincode. Fabric proves to be more efficient and scalable, due to its *execute-order-validate*[1] approach. Fabric has other pluggable features like multi-channel, endorsement, and membership policy, which motivates us to choose Fabric to replace the central curator in FL.

This paper proposes a robust and secure federated learning framework called PrivateFL, which addresses the above-discussed threats of poisoning attacks, inference attacks, and unfairness. PrivateFL replaces the central curator with a permissioned blockchain (Fabric). As compared to existing blockchain works, which primarily aim for traceability, this work aims to extract statistical insights from participants' data, while respecting their privacy. PrivateFL relies on the private data collections in Fabric, which allow an organization's peers to keep their data secure on a blockchain (or channel in Fabric) from unauthorized peers. It also facilitates multiple learning and prediction channels among different organizations and subgroups. Further, we propose a secure aggregation algorithm, which uses vertical partitioning for robust and secure aggregation. Similarly, we propose a secure prediction algorithm, which allows a third-party participant to securely query the learned model (without revealing its raw data), as well as protecting the global model.

Our key contributions are summarized as follows:

- This paper presents a robust and secure FL framework, called PrivateFL. PrivateFL integrates FL with permissioned blockchain (Hyperledger Fabric), to prevent inference attacks establishing decentralized trust and fairness in the system.
- A robust and secure aggregation scheme, called *VPSA* (Vertically Partitioned Secure Aggregation) is tailored to PrivateFL. *VPSA* uses private data collections in Fabric, to store vertically partitioned local model updates on different peers and evaluates them for quality, based on Euclidean and cosine similarity measures.
- A secure prediction mechanism is proposed for PrivateFL, which allows a third-party application to securely query the learned global model. It does the secret share of querying (raw) data, to store them on different peers, runs inference for each share, and then aggregates them to get a prediction.
- Experimental evaluations are carried out with different datasets and compared against state-of-the-art attacks and defenses to show the effectiveness of PrivateFL. Theoretical analysis proves the security of the proposed scheme.

The rest of the paper is structured as follows: Section 2 discusses the background and the related works. Section 3 discusses the threat model. Section 4 presents a fair, robust, and secure FL-enabled framework using a permissioned blockchain, called PrivateFL. Section 5 presents the theoretical analysis. Section 6 describes the experimental setup and results. Section 7 concludes this work.

2. Background and related work

This section discusses federated learning (FL), poisoning and inference attacks in FL, Hyperledger Fabric, and related work.

2.1. Federated learning

Federated Learning (FL) is a collaborative learning approach that facilitates training an intelligent model from distributed sources (McMahan et al., 2016). FL moves the model to the edge and keeps the data in place. FL comes in different flavors and settings. Vanilla FL is one of the traditional FL, which consists of a central server and multiple participants distributed across demographics. The server initiates the system by instantiating a preliminary model (training plan, w^t). Then, it broadcasts w^t to a subset (S) of the participants (N). Each participant i trains the local model (w_i^{t+1}) over its local held data (D_i) as stated in Eq. (1).

$$w_i^{t+1} \leftarrow w_i^t - \eta \nabla l(w_i^t; b \in D_i), \tag{1}$$

where η is the local learning rate of clients and b is a batch from local dataset D_i. Then they send w_i^{t+1} to the server. The server aggregates all received model updates, to get a new global model w_{t+1} by weighted averaging (FedAvg McMahan, Moore, Ramage, Hampson, & y Arcas, 2017) as stated in Eq. (2).

$$w^{t+1} \leftarrow w^t + \sum_{i=1}^{n} \frac{n_i}{\sum_{j=1}^{S} n_j} \nabla w_i^{t+1}, \tag{2}$$

where n_i is the number of private data samples with client i. The whole process runs for several iterations until convergence.

FL has two different settings: cross-device and cross-silo. Cross-device FL comprises resource-constrained devices, wherever cross-silo FL comprises resource-intensive devices. We consider a hybrid FL scenario where resource-constrained participants are associated with resource-intensive participants and collaboratively train an intelligent model in a fair, robust, and secure fashion.

2.1.1. Poisoning attacks

Federated Learning promises to be privacy-preserving in nature, but it has no control over the participant's behavior. As a result, the participants can send malicious updates to the server to degrade the global model performance. Such attacks are called poisoning attacks (Bagdasaryan et al., 2020; Cao et al., 2019; Fang et al., 2020). Poisoning attacks are mainly classified as targeted and untargeted poisoning attacks. Targeted attacks do not impact the global model performance, rather they impact a specific class of data like label flipping (Cao et al., 2019) and backdoor (Bagdasaryan et al., 2020) attacks. We can also refer to these attacks as data poisoning attacks, in which the adversary keeps the training plan intact. Untargeted attacks degrade the global model performance indiscriminately of test samples, such as local model poisoning attacks (Fang et al., 2020).

2.1.2. Inference attacks

Federated Learning ensures privacy for participants by keeping the data in place and moving the model. However, recent studies (Fredrikson et al., 2015; Luo et al., 2021; Shokri et al., 2017) have shown that adversaries do reverse engineering to infer or reconstruct sensitive data information from the participant's local model. A membership inference attack was introduced in Shokri et al. (2017), where the adversary could infer the presence of a participant in the private training data. They prepared auxiliary data and many shallow models to frame the attack model. Authors in Luo et al. (2021) discussed a feature inference attack, in which the adversary could infer global properties of the private training data from the participant's model. A model inversion attack was introduced in Fredrikson et al. (2015), where the adversary reconstructs the participant's data, given some prior non-sensitive information.

2.2. Hyperledger fabric

Hyperledger Fabric is a permissioned blockchain framework (Androulaki et al., 2018). It provides an enterprise-ready distributed ledger platform. It is governed by open-source and is established under the Linux foundation. It has a configurable and modular architecture, which makes its scope broad for different sectors. It provides many pluggable protocols. It neither requires cryptocurrency for mining nor smart contract execution. The fabric comprises a pluggable ordering service and membership service provider, an optional peer-to-peer gossip service, (smart contracts) chaincode, different ledgers, and a pluggable endorsement and validation policy. The key features of chaincode are (1) running concurrently, (2) being deployed dynamically, and (3) treating the application code as untrusted. Further, Fabric adopts the *execute-order-validate* approach for transactions, to achieve performance and scalability. The fabric has another important component, called a channel, which is a private subnet in the network to facilitate communication between any two organizations. There can be multiple channels on the network. A channel consists of multiple organizations' peers, a shared ledger, and an ordering service.

2.3. Related work

FL has been explored independently for designing a defense against data poisoning and model inference attacks to ensure fair aggregation. Different Byzantine-robust defenses (Blanchard et al., 2017; Cao et al., 2021; Yin et al., 2018) are proposed to detect insidious updates of malicious clients. They spot heterogeneity in participants' updates based on Euclidean and cosine similarity. Similarly, different inference-resistant approaches (Hao et al., 2019; Hayes & Ohrimenko, 2018; Wei et al., 2020; Xu, Li, Liu, Yang, & Lin, 2019) are also proposed in FL.

2.3.1. Poisoning defenses

A Euclidean based defense, Krum was proposed in Blanchard et al. (2017). Krum selects only one participant's update, as the global model for the next iteration. Krum computes Euclidean distance for every update to other $n - m - 2$ updates, assuming m malicious clients out of total n clients. Then, Krum calculates the sum of the squared distance between each update and the remaining (closest) $n - m - 2$ updates. In the last, the model with the minimum sum of squared distance is selected. Further, they proposed Multi-Krum, a variant of Krum. Multi-Krum selects the best-performing participants and averages them. Authors in Yin et al. (2018) proposed two robust aggregation rules, Median and Trimmed-Mean. They performed coordinate-wise filtering and aggregation. Median calculates the next global model parameters for each coordinate $k \in [d]$ as $g^k = med(x_i^k : i \in [n])$. For a system with n participants and a maximum of $\beta \in [0, \frac{1}{2})$ fraction of malicious participants, Trimmed-Mean calculates the model parameter for each coordinate $k \in [d]$ as $g^k = \frac{1}{(1-2\beta)n} \sum_{x_i^k \in S^k} x_i^k$, where S^k is obtained by removing the largest and smallest β fraction of the sorted array of all participants. A cosine-similarity based defense, FLTrust was proposed in Cao et al. (2021). FLTrust bootstraps trust by training a model on a server over a small and clean dataset. Next, the server calculates cosine similarity and norm for every update *w.r.t.* trusted root update and assigns a weighted score for final aggregation.

2.3.2. Inference defenses

Many of the existing approaches use cryptographic techniques (Hao et al., 2019; Wei et al., 2020; Xu et al., 2019) and adversarial training (Hayes & Ohrimenko, 2018) to preserve privacy of the participant's local model. Authors in Hao et al. (2019) proposed a privacy-enhanced federated learning (PEFL) scheme, using Secure Multi-Party Computation. They also considered the case of the collusion of multiple participants. A differential privacy integrated solution was proposed in Wei et al. (2020); however, it has a tradeoff with accuracy. A double-masking protocol was proposed in Xu et al. (2019) to preserve participant gradients; however, it incurs heavy communication overhead for all the participants. A differential privacy integrated solution was proposed in Wei et al. (2020); however, it has a tradeoff with accuracy. Using adversarial training, the authors in Hayes and Ohrimenko (2018) demonstrated a mitigation approach for inference attacks. However, it is computationally expensive. ShieldFL (Ma, Ma, Miao, Li, & Deng, 2022) is robust and secure which uses two-trapdoor Paillier homomorphic encryption technique. The authors considered malicious-cum-curious clients, while honest-but-curious servers in the system. However, ShieldFL is very computationally expensive, as it encrypts the model weights, element by element. A model with d parameters will produce d Paillier encryptions and ciphertexts. In addition, Paillier only encrypts integers, while floating point values (model weights) must be scaled beforehand, which inflates the encrypted model size even more. Thus, it also incurs heavy communication overhead.

2.3.3. FL with blockchain

There are some recent works on blockchain-based FL (Kasyap et al., 2022; Kasyap & Tripathy, 2021; Kim, Park, Bennis, & Kim, 2020; Korkmaz et al., 2020; Li, Shao, et al., 2021; Majeed & Hong, 2019; Weng et al., 2019; Xu & Chen, 2022). Authors in Majeed and Hong (2019) proposed a blockchain-based FL architecture called FLChain. They utilized the concept of channels in Hyperledger Fabric. They introduced the global model state trie, which is updated in the blockchain after aggregating local models received from the edge devices. A blockchain and federated learning-enabled secure architecture (Deepchain) for preserving privacy with a value-driven incentive mechanism is proposed in Weng et al. (2019). They discussed the application of FL for a distributed secure environment. A decentralized federated learning approach, ChainFL is proposed in Korkmaz et al. (2020). ChainFL delegates model storage to nodes. BlockFL (Kim et al., 2020) used a decentralized PoW blockchain network to enable verifiable and rewardable exchanging of the mobile devices' local learning model updates in FL networks. A multichannel FL architecture was coupled with permissioned blockchain for collaborative healthcare learning in Kasyap and Tripathy (2021). Authors in Li, Shao, et al. (2021) proposed blockchain-assisted decentralized federated learning, called BLADE-FL. They explored lazy participant's behavior on learning performance. A novel hierarchical IoT network fabric for decentralized federated learning (DFL) is built upon a lightweight blockchain called microchain, which is called μDFL (Xu & Chen, 2022). To ensure efficiency and privacy preservation at the network edge, each microchain network uses a hybrid Proof of Credit (PoC) block generation protocol and Voting-based Chain Finality (VCF) consensus protocol. Authors in Kasyap et al. (2022) proposed a Blockchain-assisted Reputation aware Decentralized Federated Learning Framework (BaRaDFL). BaRaDFL fosters Byzantine-robust aggregation and facilitates fair contribution-based weighted incentivization. BaRaDFL is scalable and incurs low communication overheads.

Table 1 summarizes the FL-Blockchain works, based on ten different factors and type of blockchain (Public and Private). For few works, computation and accuracy tradeoff are null, because they do not provide privacy, hence do not incur any extra computation overhead. Deepchain (Weng et al., 2019) is one of the works which provides both privacy and robustness. However, it incurs high computation overhead and does not provide storage, low latency and scalability. Additionally, it does not have any provision for secure prediction. BaRaDFL (Kasyap et al., 2022) is a scalable blockchain-based FL that facilitates Byzantine-robust aggregation, but is not privacy-preserving. FLChain Majeed and Hong (2019), PPDL (Kasyap & Tripathy, 2021), μDFL (Xu & Chen, 2022), BaRaDFL (Kasyap et al., 2022) and PrivateFL advocates for multi-channel training. PrivateFL achieves better performance, throughput compared to existing works and guarantees privacy, low accuracy tradeoff and robustness. On the top, PrivateFL incurs minimal communication and computation overhead.

Table 1
Comparison with existing works.

Frameworks	Type	Comm-unication	Comp-utation	Storage	Latency	Scalability	Multi tenancy	Byzantine robust	Privacy preserving	Secure prediction
FLchain	Public	Low	–	✗	✗	✗	✓	✗	✗	✗
Deepchain	Public	Low	High	✗	✗	✗	✗	✓	✓	✗
BlockFL	Public	Low	–	✗	✓	✗	✗	✗	✗	✗
Chain FL	Public	Low	–	✗	✗	✗	✗	✗	✗	✗
PPDL	Private	High	Low	✗	✗	✓	✓	✗	✓	✓
Blade-FL	Public	High	–	✗	✗	✗	✗	✗	✗	✗
µDFL	Private	High	–	✓	✓	✓	✓	✗	✗	✗
BaRaDFL	Public	High	–	✓	✓	✓	✓	✓	✗	✗
PrivateFL	Private	Low	Low	✓	✓	✓	✓	✓	✓	✓

Table 2
Notations.

Symbol	Description	Symbol	Description
O_i	Organization i	P	Peers in the blockchain network
U_{ij}	User or client j with organization i	D_j	Local dataset with client j
w_j	Local model of client j	$\triangle w_j$	Local model update of client j
vp_ml	Vertically partitioned shares of local model update	$\triangle w^t$	Global model update in round t
β	Number of malicious participants	br	non-IID balance rate

3. Threat model

FL is susceptible to various attacks in the presence of malicious-cum-curious clients, including poisoning and inference attacks. Besides that, the central server could also be malicious to compromise the integrity of the aggregation result and curious to learn the client's data.

Adversary's goal: An adversary (either client or the central server) may have one of the following goals:

1. The clients can frame poisoning attacks. In poisoning attacks, adversaries (malicious participants) aim to introduce malicious data into the training process to corrupt the model's learning. The adversary can intentionally modify the training data or inject false data to corrupt the model's learning process. The adversary can also inject backdoors or malicious updates into the local model parameters during the training process.
2. The central server and clients can frame inference attacks, aiming to infer sensitive information from the local model updates of other clients. Clients frame inference attacks, either by colluding with the server, or capturing data transfer in the middle.
3. The central server can perform biased aggregation, by preferably selecting a few local model updates, and dropping selective local model updates.

Adversary's knowledge: The adversary has knowledge of the proposed framework and the aggregation schemes, making them more potent for launching attacks.

Design Goals: Based on the above-discussed adversary model, our proposed framework (PrivateFL) must meet the undermentioned robustness, security and fairness requirements.

- *Robustness:* It should be Byzantine-robust and achieve no-attack accuracy even in the presence of malicious participants inducing malicious, noisy, and biased updates.
- *Security:* It should preserve confidentiality, such that none of the participants or the central server should infer sensitive information from the participant's local model update.
- *Fairness:* It is important to ensure the integrity of the aggregation results within PrivateFL. All the local model updates should be considered for robust and secure aggregation without bias.

4. PrivateFL: The proposed framework

PrivateFL provides an FL-enabled framework for providing secure learning and prediction in collaboration with different actors (FL clients). PrivateFL is integrated with permissioned blockchain, to securely perform FL operations (aggregation and prediction). PrivateFL allows multiple secure model training and sharing channels in a single shared infrastructure. Table 2 summarizes the main notations.

Components: Fig. 1 illustrates a basic infrastructure (network) to host FL using a permissioned blockchain network (Hyperledger Fabric). The components of Fabric infrastructure are described below.

- **Organizations:** Any organization can initialize a blockchain network (N). For example, organization 1 (O1) initiates a network, referred to as administrator. Orderer is the primary component when the network is formed and is configured according to network configuration (NC), defined by the administrator. Further, the administrator updates NC to share administrative rights with other collaborating organizations.
- **Certificate Authority (CA):** CA dispenses identities to each organization and its nodes. It issues *X.509* certificates to recognize the components of organizations. It is also used to sign transactions, which can be used to verify endorsement by specific organization components. There can be multiple certificate authorities for different organizations. Hyperledger Fabric provides a built-in CA, called *Fabric-CA*.
- **Membership Service Provider (MSP):** MSP is a core component that identifies the certificate properties issued by CA. NC uses MSP to identify and grant access to different players in the network. It abstracts the underlying security mechanisms for issuing, validation, and authentication.
- **Channels:** Channel is a subnetwork of the blockchain network. Every channel is defined by organizations, peers, shared ledgers, chaincodes, ordering services, and a membership service provider. Every channel initializes its ledger with a genesis block, which keeps the configuration of channel policies and associated members. It is the backbone service to support Fabric multi-tenancy.
- **Peers:** Peers are the fundamental component of a blockchain network. Different organizations in the network own and deploy peers. Peers are an integral part of the network, which connects to the organization and its associated client applications. The peer is a central point, which has a relationship to almost every other component (Channel, Orderer) in the network. A peer can join multiple channels, and host multiple chaincodes and ledgers. Peers are issued a certificate by their organization CA, which is verified by a channel MSP while joining a channel.
- **Orderer:** Orderer does transaction ordering and makes the design deterministic consensus, in contrast to probabilistic consensus in public blockchains (which causes forks). It ensures that the blocks

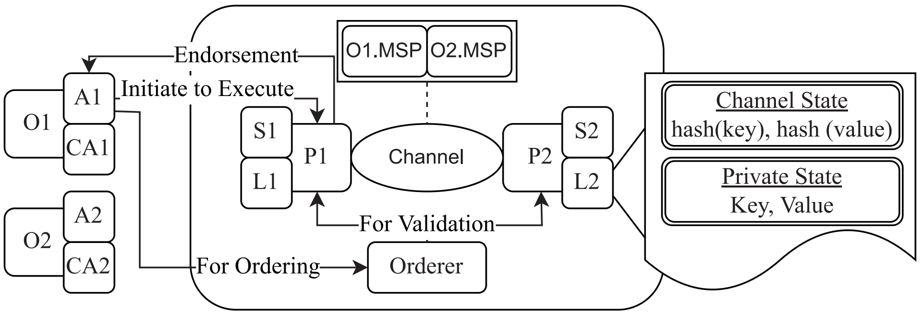


Fig. 1. Hyperledger Fabric; O: Organization, CA: Certificate Authority, MSP: Membership Service Provider, A: Client Application, P: Peer, S: Smart Contract (Chaincode), L: Ledger.

validated by peers are included in the ledger. Further, it also maintains a consortium (a list of organizations) allowed to create a channel. They come in different implementations, mainly Raft, Kafka, and Solo.

- **Smart Contract:** Smart contract, also known as chaincode in Hyperledger Fabric, is the heart from a business perspective. It defines the business logic to query from or add to the ledger (after getting invoked from an application). It defines relationships between organizations as executable code.
- **Ledger:** Ledger stores the information, contains the world state (current state) of the business, and a transaction log (blockchain). It is immutable and cannot be modified. The network consists of different ledgers and keeps them consistent through consensus. A ledger contains multiple blocks, block headers, data, transactions, metadata, and chain to the previous block's hash.
- **Private Data:** To ensure privacy and confidentiality in the system, different blockchain platforms adopted different approaches, like encrypting data and Zero Knowledge Proofs (ZKP). However, they require considerable time and resources. Fabric introduced private data[2] in its latest release. Private data allows a group of organizations to keep data private from other organizations on a channel. This could have been solved by creating different channels, but that includes additional overheads. Fabric allows peers to create private data collections (PDC), which gives the ability to endorse, commit and query private data. PDC is a collection of actual private data and a hash of the data.
- **FL Clients:** These are the client applications associated with any organization that is interested in collaboration for training an intelligent machine learning model. Further, there would be some third-party client applications, to query the intelligent model, to get predictions over their raw data.

Fig. 2 demonstrates the proposed framework, with collaborating organizations (O1, O2, and third party organizations). Every organization has its own certificate authority, peers, and membership service provider. Every organization has associated FL clients, *i.e.* , Users $U_{11} \dots U_{1n}$ associated with O1, $U_{21} \dots U_{2n}$ associated with O2. Third party organizations connect to blockchain through their own applications. Fabric supports multi-tenancy, which allows multiple channels for different purposes. Peers from collaborating organizations (O1 and O2) support two channels, Training and Inference (Prediction). Any third-party organization joins the inference channel for secure prediction. Peers host smart contracts and ledgers with multiple private data collections (PDC). The basic transaction flow for secure aggregation (steps 1–8) and prediction (steps 9–15) is described in Fig. 2.

[2] https://hyperledger-Fabric.readthedocs.io/en/release-2.2/private-data/private-data.html

4.1. Vertically partitioned secure aggregation

Here, we propose a secure aggregation scheme, which ensures fair, robust, and secure aggregation of the local model updates of FL clients. FL clients are already enrolled with their respective organization CA's. Each FL client (user i or user j owns some data D_i and D_j and trains local models w_i and w_j, respectively. Next, the FL client submits a proposal (with local model updates, $\triangle w_i$ and $\triangle w_j$) to endorsing peers of the associated organizations to invoke the aggregation chaincode. $\triangle w_i$ and $\triangle w_j$ are treated as private data and sent in the transient field of the proposal. Fig. 3 describes the private data collections (*collectionLocalModelUpdateO1*) for organization 1 (O1) and (*collectionLocalModelUpdateO2*) for organization 2 (O2) to store local model updates. *Policy* defines which organization peers can persist the private data. *requiredPeerCount* is set to 0 not to allow any distribution. *maxPeerCount* is the maximum number of other peers to distribute private data to maintain redundancy if a peer goes down. *blockToLive* tells the number of blocks that the private data should persist. *memberOnlyRead* and *memberOnlyWrite* indicate that only authorized peers are allowed to read and write.

The endorsing peers of authorized organizations do vertical partitioning of local model updates ($\triangle w_i$ and $\triangle w_j$) as described below. With m attributes (tensors) and n objects (local model updates) available, each party (node) $i \in P$ collects m_i attributes, for each object, such that $\sum_{i=1}^{P} m_i = m$. Suppose there are two local model updates ($\triangle w_i$ and $\triangle w_j$), with ten attributes (in each update) and three parties. Then the objects are distributed among 3 parties (A, B, and C) as shown below,

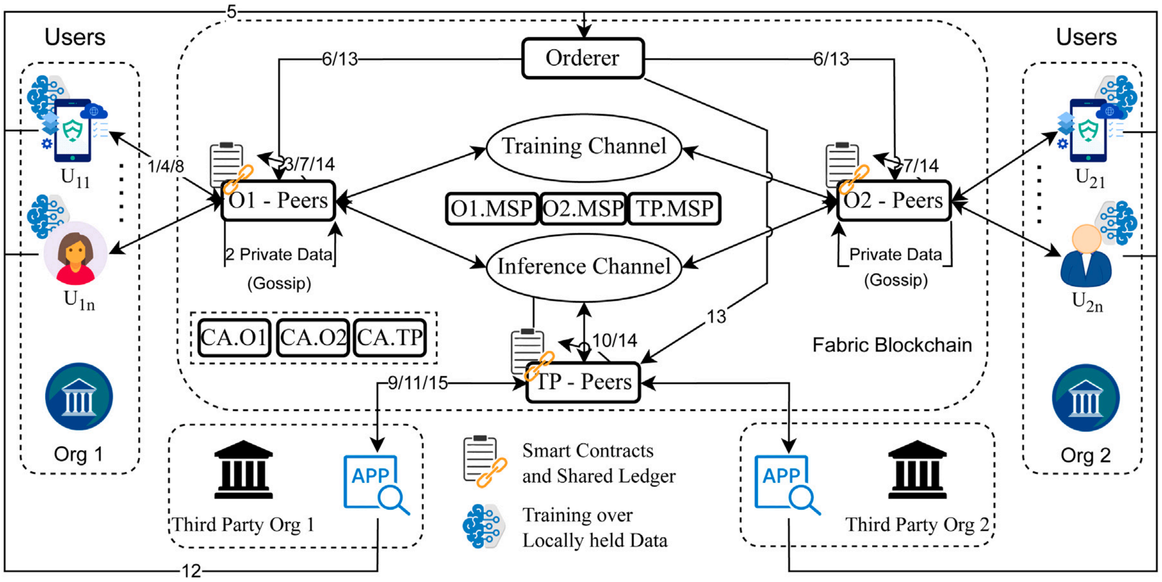
$$\triangle w_i : \begin{bmatrix} a & b & c & d & e \\ f & g & h & i & j \end{bmatrix} \rightarrow A : \begin{bmatrix} a & 0 & 0 & d & 0 \\ 0 & g & 0 & 0 & j \end{bmatrix},$$

$$B : \begin{bmatrix} 0 & b & 0 & 0 & e \\ 0 & 0 & h & 0 & 0 \end{bmatrix}, C : \begin{bmatrix} 0 & 0 & c & 0 & 0 \\ f & 0 & 0 & i & 0 \end{bmatrix}$$

where each party i collects the value at positions where $pos\%i{=}{=}0$, and always starts checking from the last index of the participant. Similarly, another update ($\triangle w_j$) is distributed among three participants.

Now, endorsing peer authorizes access to peers from different organizations by defining separate private data collections. Suppose an O1 peer receives private data (local model update $\triangle w_i$) from a user i. Peer does vertical partitioning of $\triangle w_i$ into three partitions (for O1 $\{\triangle w_{i1}\}$, O2 $\{\triangle w_{i2}\}$, and TP $\{\triangle w_{i3}\}$). Fig. 4 describes the private data collections, where O1 makes two PDC, one for O2 (*collectionLocalModelUpdateO1O2*) and another for TP (*collectionLocalModelUpdateO1TP*). Fig. 5 shows the graphical visualization, in which each ledger hosted by a peer contains the channel world state and all the defined private data collections.

Each peer from O1, O2, and TP has access to a vertically partitioned distribution of local model update of $\triangle w_i$ and $\triangle w_j$. O1 peer has $\triangle w_{i1}$ and $\triangle w_{j1}$. O2 peer has $\triangle w_{i2}$ and $\triangle w_{j2}$. TP peer has $\triangle w_{i3}$ and $\triangle w_{j3}$. Each peer is supposed to validate the vertically partitioned distributions and aggregate them. As these distributions are sparse (many fields are



Secure Aggregation	**Secure Inference**
1. FL clients associated with any organization push local model updates to the associated peer of the organization. 2. Peers of the same organization exchange local model updates (private data) through gossip protocol. 3. Peers invoke local model validation and aggregation smart contract on training channel. 4. Peers send back a response (endorsement) to FL client, with the hash of locally aggregated model of one organization, as transaction response. 5.FL clients send their endorsed transactions (without private data or local model update) to the Orderer. 6.Orderer prepares a block and sends it to all the peers across the organization for model validation. 7.Peers validate all models and perform global model aggregation. 8.Peers send back the response to the FL clients about the commit.	9. Third-Party applications send inference requests (with raw querying data) to the associated peers of their organization. 10. Peers invoke inference smart contract on another (inference) channel. 11. Peers send back the response (endorsement) to the application. 12. Application submits the endorsed transactions (without private or querying data) to the Orderer. 13. Orderer prepares a block and sends it to all the peers across the organization. 14. Peers prepare the inference result, by securely collecting the prediction logits from different peers. 15. Peers send back the inference result to the querying application.

Fig. 2. The proposed framework.

```
[                                                   [
    {                                                   {
        "name": "collectionLocalModelUpdateO1",             "name": "collectionLocalModelUpdateO2",
        "policy": "OR('O1.member')",                        "policy": "OR('O2.member')",
        "requiredPeerCount": 3,                             "requiredPeerCount": 3,
        "maxPeerCount": 3,                                  "maxPeerCount": 3,
        "blockToLive": 1000000,                             "blockToLive": 1000000,
        "memberOnlyRead": true,                             "memberOnlyRead": true,
        "memberOnlyWrite": true                             "memberOnlyWrite": true
    }                                                   }
]                                                   ]
```

Fig. 3. Private data collection (local model update).

```
[
    {
        "name": "collectionLocalModelUpdateO1O2",
        "policy": "OR('O1.member', 'O2.member')",
        "requiredPeerCount": 3,
        "maxPeerCount": 3,
        "blockToLive": 1000000,
        "memberOnlyRead": true,
        "memberOnlyWrite": true
    },
    {
        "name": "collectionLocalModelUpdateO1TP",
        "policy": "OR('O1.member', 'TP.member')",
        "requiredPeerCount": 3,
        "maxPeerCount": 3,
        "blockToLive": 1000000,
        "memberOnlyRead": true,
        "memberOnlyWrite": true
    }
]
```

Fig. 4. Private data collection (vertically partitioned).

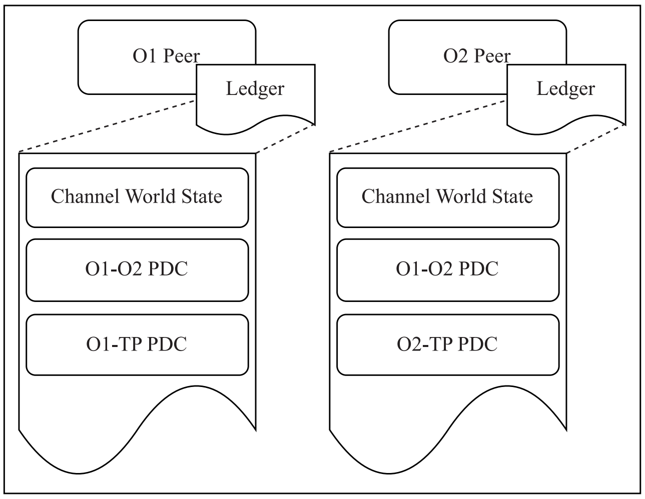


Fig. 5. Private data collections (visualization).

Algorithm 1 Vertically Partitioned Secure Aggregation (VPSA)

Input: w^t: Current global model, vp_ml: Vertically Partitioned (list of models) Distributions, β: no of malicious participants
Output: w^{t+1}: New aggregated model

```
 1: //Calculate weight for every coordinate, using both of the Euclidean and cosine weights
 2: function COORD_WEIGHTS($vp\_ml, cos\_weights, len$)
 3:     $weights \leftarrow [[0 \text{ for } k \in n] \text{ for } m \in vp\_ml]$    ▷ n: number of model parameters
 4:     $euc\_weights \leftarrow [[0 \text{ for } k \in n] \text{ for } m \in vp\_ml]$
 5:     $stacked\_vp\_ml \leftarrow \text{stack}([m \in vp\_ml], \text{dim}=0)$
 6:     $sorted\_vp\_ml \leftarrow \text{sort}(stacked\_vp\_ml)$
 7:     $min\_values \leftarrow sorted\_vp\_ml[\beta]$
 8:     $max\_values \leftarrow sorted\_vp\_ml[len - \beta]$
 9:     for each index $i$, model $m \in vp\_ml$ do
10:         for each param $k \in n$ do
11:             if $m[k] > min\_values[k]$ then
12:                 if $m[k] < max\_values[k]$ then
13:                     $euc\_weights[i][k] \leftarrow 1$
14:                 end if
15:             end if
16:             $weights[i][k] \leftarrow euc\_weights[i][k] * cos\_weights[i]$
17:         end for
18:     end for
19:     return $weights$
20: end function
21: //Calculate cosine weights
22: function COSINE_WEIGHTS($vp\_ml, w$)
23:     $cos\_weights \leftarrow [0 \text{ for } m \in vp\_ml]$
24:     for each index $i$, model $m \in vp\_ml$ do
25:         $cos\_weights[i] \leftarrow \cos(m, w)$
26:     end for
27:     return $cos\_weights$
28: end function

29: //Calculate new model parameters
30: $len \leftarrow len(vp\_ml)$
31: $cos\_weights \leftarrow cosine\_weights(vp\_ml, w)$
32: $weights \leftarrow coord\_weights(vp\_ml, cos\_weights, len)$
33: for each param $k \in n$ do
34:     $\triangle w_k^{t+1} \leftarrow \frac{\sum_{i=0}^{len} vp\_ml[i]\{k\} \times weights[i][k]}{\sum_{i=0}^{len} weights[i][k]}$
35: end for
36: $w^{t+1} \leftarrow w^t - \triangle w^{t+1}$
```

zero), Euclidean-based defense will not be effective on the complete distribution.

Therefore, we propose a customized aggregation algorithm *VPSA* (Vertically Partitioned Secure Aggregation), which assigns coordinate-wise Euclidean weights and cosine-based similarity score for complete local model update (as cosine has improved performance for sparse matrices).

Algorithm 1 describes Vertically Partitioned Secure Aggregation (VPSA). It executes at every peer, which holds the vertically partitioned distributions. For example, O1 peer has $\triangle w_{i1}$ and $\triangle w_{j1}$. The algorithm takes both these partitioned updates in a list vp_ml (vertically partitioned models), w^t (the current global model), and β (approximate number of malicious participants). First, it executes the function $cosine\_weights$ to calculate cosine weights for every sparse distribution, by calculating the cosine score with respect to w^t and returns $cos\_weights$. Next, it executes a function $coord\_weights$, which calculates $euc\_weights$ (Euclidean weights) by assigning score zero to outliers (not in range of $[\beta, len(vp\_ml) - \beta]$ sorted coordinates) for every coordinate $k \in n$. The updated coordinate weight ($weight$) is the product of $euc\_weights$ and $cos\_weights$. Finally, the new model parameters are calculated by coordinate-wise weighted averaging. Further, every peer updates the aggregated distributions to the channel world state, which is accessible to all the other peers in the channel. Then, the peers aggregate the (validated and aggregated $\{\triangle w_i\}$ from O1, $\{\triangle w_2\}$ from O2 and $\{\triangle w_3\}$ from TP) vertically partitioned updates, to get the final global model update.

Thus, VPSA achieves fair, robust, and secure aggregation, even in the presence of malicious and curious participants. None of the peers has access to complete local model updates of any other peer, rather they have only access to partitioned local model updates. The vertically partitioned shares are evaluated for their quality (robustness), which successfully removes any malicious, noisy, or biased update. The smart

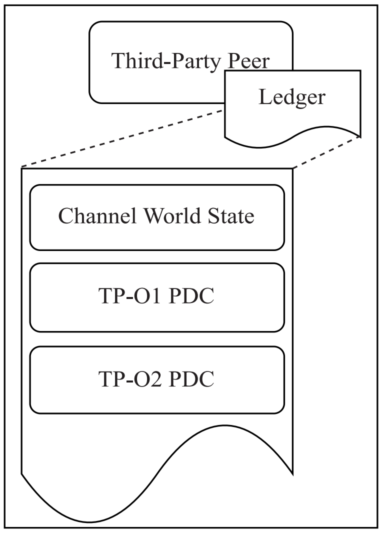


Fig. 6. Third-party ledger view.

contracts (chaincode) ensure the fair execution of all the defined rules and add decentralized trust to the system.

4.2. Third-party secure prediction

A secure prediction mechanism is proposed to securely query the global model and protect the querying data. We assume that organizations O1, O2, and Third-Party organizations have established an inference channel. Third-Party application (querying party) is already enrolled with Third-Party organization CA. The querying party owns some data D_k. It submits a proposal (with D_k) to endorsing peers of Third-Party organizations to invoke the inference chaincode. D_k is treated as private data and sent in the transient field of the proposal. Then, an authorized peer does the secret share of D_k and authorizes access to peers of training organizations by defining separate private data collections, as shown in Fig. 6. Each peer from O1 and O2 trains on secret shares of D_k. Then, the querying peer collects all the prediction logits from the accessible private data collections, as the inference result. Thus, the proposed mechanism achieves secure prediction to securely query a model. In addition, it protects the intellectual property rights of every participating organization on the training channel.

5. Theoretical analysis

We show the effectiveness of our scheme by proving Theorem 1, which is proved with the assistance of the following Lemma.

Lemma 1. *Even if m out of n model weights are known in a given neural network model, it is computationally infeasible to reconstruct the remaining $n - m$ model weights.*

Proof. The weights for each layer are independent and identically distributed (IID). The weights in each layer are independent of the weights in other layers. Also, the feature inputs are independent of the weights.

Fig. 7 illustrates a simple neural network for a yes/no classification. We can write it as below,

$$h1 = \mathcal{A}(w_{11} \times x),$$

$$h2 = \mathcal{A}(w_{12} \times x),$$

$$\hat{y} = \mathcal{A}(w_{21} \times h_1 + w_{22} \times h_2),$$

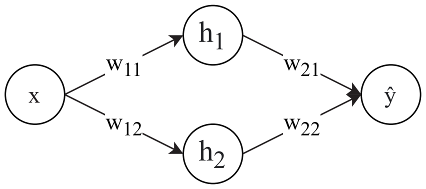


Fig. 7. A simple neural network.

where $\mathcal{A}$ is any activation (Sigmoid, Tanh, Relu) function.

Activation functions induce non-linearity, such that output at any layer is a non-linear function of the weighted sum of inputs. For example, the domain for the sigmoid activation function ($\phi(z) = \frac{1}{1+e^{-z}}$) is $(-\infty, \infty)$, and has a reduced range of (0, 1). Sigmoid is also called a squashing function and has a value of 0 for numbers less than -10 and 1 for greater than 10. It causes a loss of information for most of the values. Similarly, Tanh is also like a logistic sigmoid with a range (-1 to 1), which causes a loss of information. Relu function squashes all the values less than 0, using the $max(0, z)$ function over the set of 0.0 and the input z. So, for half of the domain $(-\infty, 0)$, the information is lost, when using the Relu activation function.

Taking strong assumption that, an adversary knows w_{11}, w_{12} and w_{21} along with $\hat{y}$, it is difficult to reconstruct w_{22}. Thus, solving the equation for w_{22} (even if knowing $\mathcal{A}(w_{21} \times h_1 + w_{22} \times h_2)$, w_{21}, h_1 and h_2) becomes computationally infeasible, as there can be infinite solutions for w_{22} due to the confusion engendered by the activation function $\mathcal{A}$.

We take stronger assumption that, an adversary knows w_{11}, w_{12} and w_{21} along with y (true prediction) and $\hat{y}$ (model prediction). Then, the adversary runs backpropagation to update w_{22} as,

$$w_{22} = w_{22} - \eta \times \frac{\partial \mathcal{J}}{\partial w_{22}},$$

where η is the learning rate. $\mathcal{J}$ is the prediction error $(y - \hat{y})$, which is further derived as $y - \mathcal{A}(w_{21} \times h_1 + w_{22} \times h_2)$. However, solving for $\frac{\partial \mathcal{J}}{\partial w_{22}}$ for w_{22} is still computationally infeasible, due to the loss of information caused by the activation function $\mathcal{A}$ and non-monotonicity of differentiable functions. □

Theorem 1. *VPSA achieves secure aggregation, even if $n-1$ peers compromise and exchange their model (vertically partitioned) shares. The compromised participants would fail to reconstruct the remaining share(s).*

Proof. For brevity, it is solved for a model weight matrix of 2×5 dimensions and can be generalized for $n \times m$ dimensions.

Assume w_1 is a weight matrix owned by peer P_1.

$$w_1 : \begin{bmatrix} w_{11} & w_{12} & w_{13} & w_{14} & w_{15} \\ w_{21} & w_{22} & w_{23} & w_{24} & w_{25} \end{bmatrix},$$

Let, P_1 makes three shares of it, such that P_2 and P_3 have the following vertically partitioned shares,

$$P2 : \begin{bmatrix} 0 & w_{12} & 0 & 0 & w_{15} \\ 0 & 0 & w_{23} & 0 & 0 \end{bmatrix},$$

$$P3 : \begin{bmatrix} 0 & 0 & w_{13} & 0 & 0 \\ w_{21} & 0 & 0 & w_{24} & 0 \end{bmatrix},$$

So, P_2 and P_3 together have w'_1,

$$w'_1 : \begin{bmatrix} 0 & w_{12} & w_{13} & 0 & w_{15} \\ w_{21} & 0 & w_{23} & w_{24} & 0 \end{bmatrix}.$$

Assume P_2 and P_3 have knowledge of an input (x, y) pair. Therefore, they can compute $\hat{y} = \mathcal{A}(w'_1 \times x)$. They can also compute $\mathcal{J}$: $(y - \mathcal{A}(w'_1 \times x))$. Now, based on Lemma 1, solving for remaining values $(w_{11}, w_{14}, w_{22}, w_{25})$ is computationally infeasible due to the loss of information induced by non-linearity (brought by the activation function $\mathcal{A}$ in any neural network).

It proves that, *VPSA* achieves secure aggregation, even in the presence of $n - 1$ compromised peers. □

6. Experiments and results

The implementation of PrivateFL's prototype is done with multiple organizations (O1, O2 and third-party organizations). O1 initiates the network. Further, O1 adds other organizations (O2 and Third-Party organizations). Fabric offers a number of APIs to interact with client applications. PrivateFL client (web) applications are written in Python (Flask and PyTorch), and interact with Fabric using Node SDK. PrivateFL uses Apache Kafka as the ordering service, where each channel consists of a single partition topic. Kafka (Broker and Zookeeper) is hosted using docker on a server machine with processor Intel(R) Xeon(R) CPU E5-2650 v4 @ 2.20 GHz and 256 GB RAM.

Evaluation datasets: We considered three datasets for evaluation: MNIST, Fashion-MNIST and CIFAR-10 datasets. MNIST is a dataset of handwritten digits with 70,000 images, equally distributed in 0–9. Fashion-MNIST is a dataset of grayscale fashion products, with 60,000 training and 10,000 test images, equally distributed in 10 classes. Each sample in MNIST and Fashion-MNIST is of size 28×28. CIFAR-10 is a complex dataset consisting, of 60000 32×32 color images.

Learning classifiers: We used a lightweight convolutional neural network for training MNIST and Fashion-MNIST, with 32 and 64 kernels of size 3×3, followed by a max-pooling layer and two fully connected layers with, 9216 and 128 neurons. ReLU activation is used in each layer with a dropout of 0.25. ResNet-101 is used to train CIFAR-10.

FL settings and parameters: We considered 20 participants. Each participant trains locally for 1 round with a batch size of 32. The whole process iterates for 20 rounds. For carrying out poisoning attacks, 5%, 10%, and 20% of the participants are considered to be malicious. The attack is continuous and executed in all rounds. It is assumed that the data distribution is independent and identically distributed (IID), unless it is specified. In addition, we evaluated the impact on non-IID settings by varying the balance rate (br=0.25,0.50,0.75), which is the ratio between a class's data volume and its counterpart (Li, Sun, et al., 2021).

Evaluated attacks: We considered the state-of-the-art data poisoning, *i.e.,* backdoor (targeted), and local model poisoning (untargeted) attacks for comparison and evaluation.

Backdoor attack: The adversary poisons the training data with certain features to get misclassified to another target label (Bagdasaryan et al., 2020). The resultant model predicts the wrong label in the presence of these features.

Fang attack: A local model poisoning attack, which finds a malicious model update by solving an optimization objective to deviate from the benign local model updates in the opposite direction (Fang et al., 2020).

Poisoning defenses: We compared the proposed defense (*VPSA*) with Trimmed-Mean (Yin et al., 2018), FLTrust (Cao et al., 2021) and FedAvg (McMahan et al., 2017). FedAvg is a weighted (no-defense) aggregation rule, which simply averages the local model updates. Trimmed-Mean is a coordinate-wise Byzantine-robust aggregation rule, which filters out malicious coordinates based on Euclidean distance. FLTrust is a cosine-based Byzantine-robust aggregation rule, which assigns a trust score based on cosine similarity to a root trust update, followed by weighted averaging.

Measurement metrics: We evaluated backdoor attack in terms of backdoor success rate, *i.e.,* the percentage of images (embedded with trigger) classified to the target label. Fang attack is evaluated in terms of accuracy drop. Traditional and FL-enabled frameworks are compared in terms of accuracy. The efficiency of Kafka is measured in terms of throughput and request latency.

Table 3
Backdoor success rate against Data Poisoning (Backdoor) Attack.

Datasets	%m	FedAvg	TMean	FLTrust	*VPSA*
MNIST	5	4.56	2.47	3.82	0.16
	10	5.84	3.52	4.67	0.19
	20	6.38	4.61	5.12	0.22
Fashion-MNIST	5	5.92	3.15	3.92	0.21
	10	6.18	4.32	4.76	0.22
	20	7.97	5.67	5.86	0.25
CIFAR-10	5	15.42	5.61	4.16	1.41
	10	18.68	7.27	5.82	1.87
	20	19.36	8.24	7.27	1.92

Table 4
Attack impact (loss in accuracy) against Local Model Poisoning (Fang) Attack.

Datasets	br	FedAvg	TMean	FLTrust	*VPSA*
MNIST	5	8.32	4.52	2.24	1.16
	10	10.14	5.62	2.86	1.27
	20	11.52	7.84	3.15	1.42
Fashion-MNIST	5	12.36	5.14	2.56	1.36
	10	13.98	5.72	3.72	1.41
	20	14.18	6.13	4.16	1.48
CIFAR-10	5	18.82	7.24	3.14	1.46
	10	19.74	8.51	5.18	1.81
	20	21.46	9.63	6.69	1.98

6.1. Results

Attack Evaluation: Table 3 logs the backdoor success rate of *VPSA* in the presence of 5%, 10%, and 20% malicious participants. *VPSA* restricts the backdoor success rate up to 0.22% over the MNIST dataset and 0.25% over the Fashion-MNIST dataset. TMean and FLTrust still caused around 5% backdoor success. Table 4 logs the attack impact (loss in accuracy) against fang attack. It can be observed that *VPSA* restricted the attack impact up to 1.42% and 1.48% compared to no-attack accuracy of 97.62% (MNIST) and 87.8% (Fashion-MNIST) respectively.

Attack impact on larger datasets with complex models: VPSA is evaluated for robustness against backdoor (data poisoning) and fang (local model poisoning) attack, in the presence of 5%, 10% and 20% malicious participants, over CIFAR-10 dataset with complex model architecture ResNet-101. The no-attack accuracy over CIFAR-10 is 79.85%. *VPSA* restricted backdoor success rate to 1.41 in the presence of 5% attackers, and performed comparatively better than Trimmed Mean and FLTrust. *VPSA* restricted backdoor success rate to 2% in the presence of 10% and 20% attackers. *VPSA* restricted the attack impact (loss in accuracy) against Fang (local model poisoning) attack to 1.81%, compared to no-attack accuracy.

Impact of heterogeneity: We evaluated *VPSA*, by simulating federated learning with different non-IID training data distributions. The attacks are evaluated with varying balance rates (br = 0.25, 0.50 and 0.75). Table 5 logs the attack impact over the CIFAR-10 dataset, considering 20% attackers. It can be observed that *VPSA* restricted the backdoor success rate and attack impact (loss in accuracy) against data and local model poisoning attacks, respectively. The backdoor success rate is restricted to 0.25% while the maximum loss in accuracy caused by a Fang attack is less than 1.5% compared to no-attack accuracy.

Time: Table 6 shows the time taken by different aggregation schemes. *VPSA* consumes 1.32 s, which is comparable to FedAvg. However, TMean consumes more time, due to coordinate-wise operation and FLTrust due to normalization operation. Further, PrivateFL takes negligible time for vertical partitioning and sharing, while state-of-the-art inference-resistant approaches using cryptographic algorithms take more than 2000 s to encrypt and 60 s to decrypt using Paillier (Homomorphic Encryption) cryptosystem. Additionally, it generates

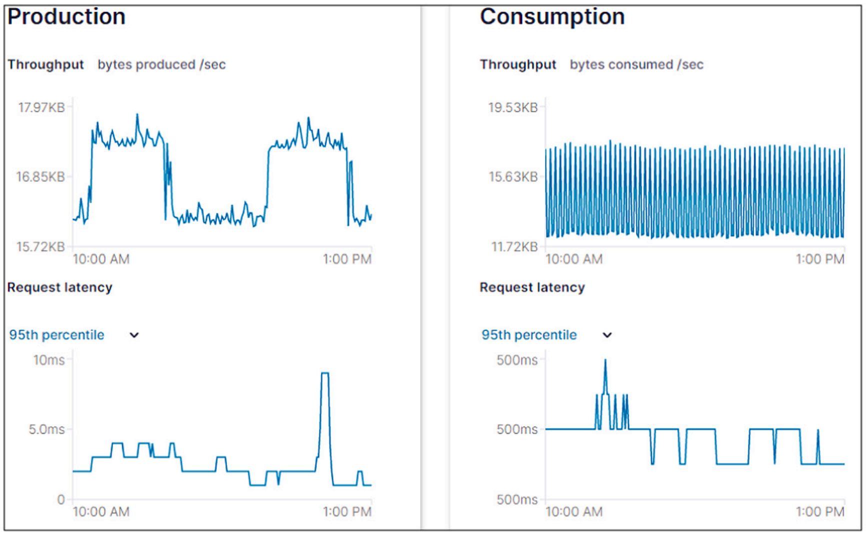


Fig. 8. PrivateFL: Performance metrics (95th percentile means, 95% of times it behaves like this.).

Table 5
Model performance by varying balance rate (*br*) under non-IID FL setting over CIFAR-10 dataset.

Attack	*br*	FedAvg	TMean	FLTrust	*VPSA*
Backdoor	0.25	6.24	1.23	2.58	0.24
	0.50	5.46	1.34	2.76	0.18
	0.75	5.17	1.41	2.54	0.21
Fang	0.25	14.26	8.24	11.12	1.48
	0.50	13.69	7.36	10.04	1.16
	0.75	13.97	8.07	10.48	1.27

Table 6
Comparison of aggregation time.

Scheme	FedAvg	TMean	FLTrust	*VPSA*
Time (in s)	1.06	5.4	27	1.32

larger ciphertext and incurs heavy communication overheads. Similarly, when Multi-Party Computing (MPC) based solution is used on the participant's side, then the participants incur heavy communication overheads. Both approaches are not even compatible with the existing Byzantine-robust aggregation schemes.

Performance Analysis: The proposed framework is efficient and scalable, utilizing the *execute-order-validate* approach of Fabric. It also achieves high throughput and low latency, as the time difference between executing vanilla FL and using Fabric is only 0.2 s. In order to achieve real-time and efficient communication, different topologies can be considered, such as centralized, distributed, cyclic and acyclic. Here, an acyclic all-reduce distributed topology is considered. Each (producer) participant publishes to one topic, which is subscribed to by every consumer (participant). A broker acts as a server that hosts topics. The number of brokers and partitions in a topic can be increased for load balancing. Kafka is a highly scalable (high throughput and low latency), fault-tolerant, distributed message processing framework used for storing and sharing models. Fig. 8 shows Kafka broker performance for producer/consumer throughput and request latency. Producer throughput is more stable (with low request latency) due to a faster rate of exchange processing the messages than producers producing and sending messages. Consumers consume messages faster than producers are able to publish messages to the exchanges. As a result, producer throughput is a ceiling for consumer throughput; therefore, frequent spikes are observed in consumer throughput and also have high request latency.

7. Conclusion

This paper presented a secure FL-enabled framework, called PrivateFL. PrivateFL is coupled with permissioned blockchain, Hyperledger Fabric. It provided a secure data/model sharing platform to enrich knowledge (data sharing) by incorporating every possible individual and device across demographics. The proposed secure aggregation algorithm *VPSA* (Vertically Partitioned Secure Aggregation) achieved fair, robust, and secure aggregation in the presence of multiple (malicious and curious) participants. Theoretical analysis proved that *VPSA* achieved secure aggregation. Further, this paper proposed a secure prediction mechanism, which allowed third-party participants to securely query a global model. PrivateFL used private data in Fabric with vertically partitioned computing to ensure privacy and confidentiality. PrivateFL facilitated multi-tenancy to run multiple channels for training different tasks. Empirical evaluation justified that the proposed framework achieved the above-mentioned security goals.

CRediT authorship contribution statement

Harsh Kasyap: Conceptualization, Investigation, Methodology, Experiments, Writing. **Somanath Tripathy:** Conceptualization, Investigation, Methodology, Review.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgment

We acknowledge the Ministry of Education, Government of India, for providing fellowship to complete this work.

References

Androulaki, E., Barger, A., Bortnikov, V., Cachin, C., Christidis, K., De Caro, A., et al. (2018). Hyperledger fabric: A distributed operating system for permissioned blockchains. In *Proceedings of the thirteenth EuroSys conference* (pp. 1–15).

Bagdasaryan, E., Veit, A., Hua, Y., Estrin, D., & Shmatikov, V. (2020). How to backdoor federated learning. In *International conference on artificial intelligence and statistics* (pp. 2938–2948). PMLR.

Blanchard, P., El Mhamdi, E. M., Guerraoui, R., & Stainer, J. (2017). Machine learning with adversaries: Byzantine tolerant gradient descent. In *Proceedings of the 31st international conference on neural information processing systems* (pp. 118–128).

Cao, D., Chang, S., Lin, Z., Liu, G., & Sun, D. (2019). Understanding distributed poisoning attack in federated learning. In *2019 IEEE 25th international conference on parallel and distributed systems* (pp. 233–239). IEEE.

Cao, X., Fang, M., Liu, J., & Gong, N. (2021). FLTrust: Byzantine-robust Federated Learning via Trust Bootstrapping. In *NDSS*.

European Commission (2016). Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation) (Text with EEA relevance).

Fang, M., Cao, X., Jia, J., & Gong, N. (2020). Local model poisoning attacks to byzantine-robust federated learning. In *29th {USENIX} security symposium* (pp. 1605–1622).

Fredrikson, M., Jha, S., & Ristenpart, T. (2015). Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security* (pp. 1322–1333).

Hao, M., Li, H., Luo, X., Xu, G., Yang, H., & Liu, S. (2019). Efficient and privacy-enhanced federated learning for industrial artificial intelligence. *IEEE Transactions on Industrial Informatics*, *16*(10), 6532–6542.

Hayes, J., & Ohrimenko, O. (2018). Contamination attacks and mitigation in multi-party machine learning. In *Advances in neural information processing systems. Vol. 31*.

Kasyap, H., Manna, A., & Tripathy, S. (2022). An efficient blockchain assisted reputation aware decentralized federated learning framework. *IEEE Transactions on Network and Service Management*.

Kasyap, H., & Tripathy, S. (2021). Privacy-preserving decentralized learning framework for healthcare system. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, *17*(2s), 1–24.

Kasyap, H., & Tripathy, S. (2022). Hidden vulnerabilities in cosine similarity based poisoning defense. In *2022 56th Annual conference on information sciences and systems* (pp. 263–268). IEEE.

Kasyap, H., & Tripathy, S. (2024). Beyond data poisoning in federated learning. *Expert Systems with Applications*, *235*, Article 121192.

Kim, H., Park, J., Bennis, M., & Kim, S.-L. (2020). Blockchained on-device federated learning. *IEEE Communications Letters*, *24*(6), 1279–1283.

Kong, X., Gao, H., Shen, G., Duan, G., & Das, S. K. (2021). FedVCP: A federated-learning-based cooperative positioning scheme for social internet of vehicles. *IEEE Transactions on Computational Social Systems*.

Korkmaz, C., Kocas, H. E., Uysal, A., Masry, A., Ozkasap, O., & Akgun, B. (2020). Chain FL: decentralized federated machine learning via blockchain. In *2020 Second international conference on blockchain computing and applications* (pp. 140–146). IEEE.

Kulkarni, P. P., Kasyap, H., & Tripathy, S. (2021). DNet: An efficient privacy-preserving distributed learning framework for healthcare systems. In *International conference on distributed computing and internet technology* (pp. 145–159). Springer.

Li, J., Shao, Y., Wei, K., Ding, M., Ma, C., Shi, L., et al. (2021). Blockchain assisted decentralized federated learning (BLADE-FL): Performance analysis and resource allocation. *IEEE Transactions on Parallel and Distributed Systems*, *33*(10), 2401–2415.

Li, A., Sun, J., Wang, B., Duan, L., Li, S., Chen, Y., et al. (2021). LotteryFL: Empower edge intelligence with personalized and communication-efficient federated learning. In *2021 IEEE/ACM symposium on edge computing* (pp. 68–79).

Lu, Y., Huang, X., Dai, Y., Maharjan, S., & Zhang, Y. (2019). Blockchain and federated learning for privacy-preserved data sharing in industrial IoT. *IEEE Transactions on Industrial Informatics*, *16*(6), 4177–4186.

Luo, X., Wu, Y., Xiao, X., & Ooi, B. C. (2021). Feature inference attack on model predictions in vertical federated learning. In *2021 IEEE 37th international conference on data engineering* (pp. 181–192). IEEE.

Ma, Z., Ma, J., Miao, Y., Li, Y., & Deng, R. H. (2022). ShieldFL: Mitigating model poisoning attacks in privacy-preserving federated learning. *IEEE Transactions on Information Forensics and Security*, *17*, 1639–1654. http://dx.doi.org/10.1109/TIFS.2022.3169918.

Majeed, U., & Hong, C. S. (2019). Flchain: Federated learning via MEC-enabled blockchain network. In *2019 20th Asia-pacific network operations and management symposium* (pp. 1–4).

Manna, A., Kasyap, H., & Tripathy, S. (2021). Moat: Model agnostic defense against targeted poisoning attacks in federated learning. In *International conference on information and communications security* (pp. 38–55). Springer.

Manna, D., Kasyap, H., & Tripathy, S. (2022). MILSA: Model interpretation based label sniffing attack in federated learning. In *Information systems security: 18th International conference, ICISS 2022, Tirupati, India, December 16–20, 2022, Proceedings* (pp. 139–154). Springer.

McMahan, H. B., Moore, E., Ramage, D., & y Arcas, B. A. (2016). Federated learning of deep networks using model averaging. CoRR. arXiv:1602.05629.

McMahan, B., Moore, E., Ramage, D., Hampson, S., & y Arcas, B. A. (2017). Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics* (pp. 1273–1282). PMLR.

Nguyen, D. C., Ding, M., Pham, Q.-V., Pathirana, P. N., Le, L. B., Seneviratne, A., et al. (2021). Federated learning meets blockchain in edge computing: Opportunities and challenges. *IEEE Internet of Things Journal*, *8*(16), 12806–12825.

Shokri, R., Stronati, M., Song, C., & Shmatikov, V. (2017). Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy* (pp. 3–18). IEEE.

Singh, N., Kasyap, H., & Tripathy, S. (2020). Collaborative learning based effective malware detection system. In *Joint European conference on machine learning and knowledge discovery in databases* (pp. 205–219). Springer.

Wang, X., Garg, S., Lin, H., Kaddoum, G., Hu, J., & Hassan, M. M. (2023). Heterogeneous blockchain and AI-driven hierarchical trust evaluation for 5G-enabled intelligent transportation systems. *IEEE Transactions on Intelligent Transportation Systems*, *24*(2), 2074–2083. http://dx.doi.org/10.1109/TITS.2021.3129417.

Wang, X., Garg, S., Lin, H., Kaddoum, G., Hu, J., & Hossain, M. S. (2022). A secure data aggregation strategy in edge computing and blockchain-empowered internet of things. *IEEE Internet of Things Journal*, *9*(16), 14237–14246. http://dx.doi.org/10.1109/JIOT.2020.3023588.

Wang, X., Hu, J., Lin, H., Garg, S., Kaddoum, G., Piran, M. J., et al. (2022). QoS and privacy-aware routing for 5G-enabled industrial internet of things: A federated reinforcement learning approach. *IEEE Transactions on Industrial Informatics*, *18*(6), 4189–4197. http://dx.doi.org/10.1109/TII.2021.3124848.

Wei, K., Li, J., Ding, M., Ma, C., Yang, H. H., Farokhi, F., et al. (2020). Federated learning with differential privacy: Algorithms and performance analysis. *IEEE Transactions on Information Forensics and Security*, *15*, 3454–3469.

Weng, J., Weng, J., Zhang, J., Li, M., Zhang, Y., & Luo, W. (2019). Deepchain: Auditable and privacy-preserving deep learning with blockchain-based incentive. *IEEE Transactions on Dependable and Secure Computing*, *18*(5), 2438–2455.

Wood, G., et al. (2014). Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Project Yellow Paper*, *151*(2014), 1–32.

Xu, R., & Chen, Y. (2022). μDFL: A secure microchained decentralized federated learning fabric atop IoT networks. *IEEE Transactions on Network and Service Management*.

Xu, G., Li, H., Liu, S., Yang, K., & Lin, X. (2019). Verifynet: Secure and verifiable federated learning. *IEEE Transactions on Information Forensics and Security*, *15*.

Yin, D., Chen, Y., Kannan, R., & Bartlett, P. (2018). Byzantine-robust distributed learning: Towards optimal statistical rates. In *International conference on machine learning* (pp. 5650–5659). PMLR.