This article has been accepted for publication in IEEE Transactions on Services Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TSC.2023.3318990

B^2SFL: A Bi-level Blockchained Architecture for Secure Federated Learning-based Traffic Prediction

Hao Guo, *Member, IEEE,* Collin Meese, *Student Member, IEEE,* Wanxin Li*, *Member, IEEE,* Chien-Chung Shen, *Member, IEEE,* and Mark Nejad, *Member, IEEE*

***Abstract*—Federated Learning (FL) is a privacy-preserving machine learning (ML) technology that enables collaborative training and learning of a global ML model based on aggregating distributed local model updates. However, security and privacy guarantees could be compromised due to malicious participants and the centralized FL server. This paper proposed a bi-level blockchained architecture for secure federated learning-based traffic prediction. The bottom and top layer blockchain store the local model and global aggregated parameters accordingly, and the distributed homomorphic-encrypted federated averaging (DHFA) scheme addresses the secure computation problems. We propose the partial private key distribution protocol and a partially homomorphic encryption/decryption scheme to achieve the distributed privacy-preserving federated averaging model. We conduct extensive experiments to measure the running time of DHFA operations, quantify the read and write performance of the blockchain network, and elucidate the impacts of varying regional group sizes and model complexities on the resulting prediction accuracy for the online traffic flow prediction task. The results indicate that the proposed system can facilitate secure and decentralized federated learning for real-world traffic prediction tasks.**

***Index Terms*—Blockchain, Federated Learning, Traffic Prediction, Secure Averaging, Homomorphic Encryption.**

I. Introduction

Traffic prediction plays a significant role in improving the utilization of traffic network capacity while also helping alleviate congestion by empowering traffic management centers (TMCs) and road operators to control traffic more effectively. In addition, other applications such as route guidance and navigation systems can leverage traffic prediction methods to provide travelers with more accurate information in real time.

Despite their strong performance in the literature, state-of-the-art traffic prediction models (e.g., deep learning (DL) models) can only perform well if trained using centralized big traffic data [1]. However, centralized approaches are becoming unsuitable for emerging intelligent transportation systems (ITS) where data collection is decentralized, dynamic, and performed by heterogeneous devices (e.g., sensors, mobile phones, connected vehicles). In contrast to centralized methods, collaborative and decentralized ML approaches can better match the ITS environment, allowing models to be trained and updated online directly at the network edge to improve response time and modeling efficiency for critical ITS applications.

Recently, Federated learning (FL) has shown promise in facilitating privacy-preserving and collaborative machine learning across multiple application domains (e.g., healthcare [2], internet of things (IoT) [3], and transportation [4]). During the FL process for traffic flow prediction, devices use their locally stored data to train location-specific traffic flow prediction models, which are periodically merged to generate a global model from the contributions of each participant. Consequently, FL can train transportation models decentrally, without necessitating data sharing, protecting the privacy of each participant's data. For these reasons, FL is a strong candidate for improving traffic prediction for emerging ITS.

Although the existing literature on FL-based traffic prediction is sparse, some work has experimented with FL for traffic flow prediction models [4], [5]. However, the existing work generally focuses on the large-scale macro FL case, where significant volumes of historical data are collected by a small number of participating organizations possessing sufficient processing power. Moreover, existing experimental setups appear to simulate FL using identical data shards across participants. This scenario is not practical given the increasing decentralization of ITS, where participants are heterogeneous with respect to data volumes and computing resources and collect location-specific data sets. The reader is referred to our recent critical review on traffic prediction methods [6] for more details.

Most importantly, a fundamental challenge in fully decentralized federated learning is to prevent the client from reconstructing the private data of another client from its shared updates while maintaining a good level of utility for

Manuscript received December 2, 2022; revised June 29, 2023; accepted September 20, 2023. This work is partially supported by the Guangdong Basic and Applied Basic Research Foundation under the Grant No. 2021A1515110286 from 2021-2024, the Basic Research Programs of Taicang under Grant No. TC2022JC23, and Natural Science Foundation of Shaanxi Provincial Department of Education under Grant No. 2022JQ-639, and a Federal Highway Administration grant: "Artificial Intelligence Enhanced Integrated Transportation Management System", 2020-2023. *(Corresponding author: Wanxin Li.)*

Hao Guo is with the Research & Development Institute of Northwestern Polytechnical University in Shenzhen, 518057, China (e-mail: haoguo@nwpu.edu.cn).

Wanxin Li is with the Department of Communications and Networking, Xi'an Jiaotong-Liverpool University, Suzhou, 215123, China (e-mail: wanxin.li@xjtlu.edu.cn).

Collin Meese and Mark Nejad are with the Department of Civil and Environmental Engineering, University of Delaware, Newark, Delaware, 19716, USA (e-mail: {cmeese, nejad}@udel.edu).

Chien-Chung Shen is with the Department of Computer and Information Sciences, University of Delaware, Newark, Delaware, 19716, USA (e-mail: cshen@udel.edu).

Authorized licensed use limited to: Xi'an Jiaotong-Liverpool University. Downloaded on October 05,2023 at 08:02:57 UTC from IEEE Xplore. Restrictions apply.
© 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Services Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TSC.2023.3318990

the learned models [7]. Unfortunately, such local privacy approaches often have a high cost in utility and do not easily integrate with fully decentralized algorithms [7]. To address these existing drawbacks, homomorphic encryption [8] could be a potential solution. Compared to other cryptography techniques, homomorphic encryption is a form of encryption that permits users to perform computations on its encrypted data without decrypting it [8], [9]. Homomorphic encryption can be used for privacy-preserving outsourced storage and computation [9]. It allows data to be encrypted and outsourced to a federated learning server or other computing devices. The challenge is how to integrate homomorphic encryption with federated learning procedures to preserve privacy and secure computation.

This paper proposes the B^2SFL, a bi-level blockchained architecture for secure federated learning-based traffic prediction. To address the different geographic location properties of intelligent vehicles and roadway sensors and the problems of data integrity, traceability, and system scalability, the bottom layer blockchain serves as the ledger to record the local model parameters generated from the roadside edge nodes (REN) and provide the forensics for global averaging parameters. To address the secure computation problems when the federated learning server aggregates device updates in the traditional FL procedure, we propose the DHFA (distributed homomorphic-encrypted federated averaging) algorithm and store the averaged global model parameters in the top layer blockchain to provide data security and privacy.

In summary, this paper makes the following contributions:

- We proposed a novel bi-level blockchained architecture for secure federated learning-based traffic prediction. The REN and bottom layer blockchain conduct the local model training process and store the local model parameters. The top layer blockchain stores the DHFA (Distributed Homomorphic-encrypted Federated Averaging) protected global model parameters for all regions.
- The new DHFA algorithm addresses the security and privacy computation problems in the parameter aggregation procedure for federated learning. In particular, we design the new partial private key distribution protocol and a partial encryption/decryption scheme to achieve end-to-end privacy-preserving features.
- We implemented the proposed architecture using Hyperledger Fabric, Jspaillier library, and Google Colab platform. Experimental results indicate that the system has efficient data encryption/decryption time and elucidate the impacts of regional sensor groupings on prediction accuracy for online FL models. Additionally, blockchain experiments demonstrate that transaction throughputs and latencies are suitable for real-world deployment.

The remainder of the paper is organized as follows. We discuss related work in Section II. In Section III, we describe the system architecture. In addition, we present the problem statement, permissoned blockchain network, online federated learning of traffic prediction, DHFA protocol, and workflow of B^2SFL. In Section IV, we conduct a system analysis. We discussed the security analysis and threat model in Section V. Experiments and evaluations are presented in Section VI. Section VII concludes the paper and points out future research directions.

II. Related Work

We review state-of-art research work on blockchain-based federated learning schemes for IoT applications. Chai et al. [10] described a hierarchical blockchain framework and a hierarchical federated learning algorithm for knowledge sharing. Vehicles learn environmental data through machine learning methods and share the learned knowledge. Jia et al. [11] proposed a blockchain-enabled federated learning data protection aggregation scheme with differential privacy and homomorphic encryption in an IoT environment. However, they do not consider the decentralized deployment of federated learning scheme.

Shayan et al. [12] proposed Biscotti: a fully decentralized P2P approach to multi-party ML, which utilizes blockchain and cryptographic primitives to coordinate a privacy-preserving ML process between peering clients. Lu et al. [13] proposed the blockchain-empowered asynchronous federated learning for secure data sharing on the Internet of Vehicles. They developed a hybrid blockchain architecture that consists of the permissioned blockchain and local Directed Acyclic Graph (DAG) to enhance the reliability and security of model parameters. Peng et al. [14] proposed a verifiable and auditable federated learning framework based on the blockchain system. However, the blockchain system performance evaluation is missing in all proposed schemes.

Feng et al. [15] proposed BAFL: A blockchain-based asynchronous federated learning framework. The blockchain ensures that model data cannot be tampered with, while asynchronous learning speeds up global aggregation. Li et al. [16] proposed a decentralized FL framework by integrating blockchain into FL. Every client broadcasts its trained model to other clients, aggregates the model with received ones, and competes to generate a block before its local training in the next round. Qu et al. [17] developed a decentralized paradigm for big data-driven cognitive computing (D2C), using federated learning and blockchain jointly. However, they do not consider the internal privacy problem of federated learning.

Qi et al. [18] proposed blockchain-based federated learning (BFL) with a reputation mechanism for model aggregation. A reputation-constrained data contribution and reward allocation mechanism encourages data owners to participate in BFL and contribute high-quality data. Mothukuri et al. [19] proposed a blockchain-in-the-loop FL approach that combines classic FL and Hyperledger Fabric with a gamification component. Liu et al. [20] proposed a blockchain and federated learning model for collaborative intrusion detection in vehicular edge computing. It analyzed common attacks and shows that the proposed scheme achieves cooperative privacy-preservation for vehicles.

Zhao et al. [21] proposed the blockchain-based federated learning method that preserves privacy for IoT devices. They enforced differential privacy on the extracted features to protect customers' privacy and improve the test accuracy. Qi

Authorized licensed use limited to: Xi'an Jiaotong-Liverpool University. Downloaded on October 05,2023 at 08:02:57 UTC from IEEE Xplore. Restrictions apply.
© 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Services Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TSC.2023.3318990

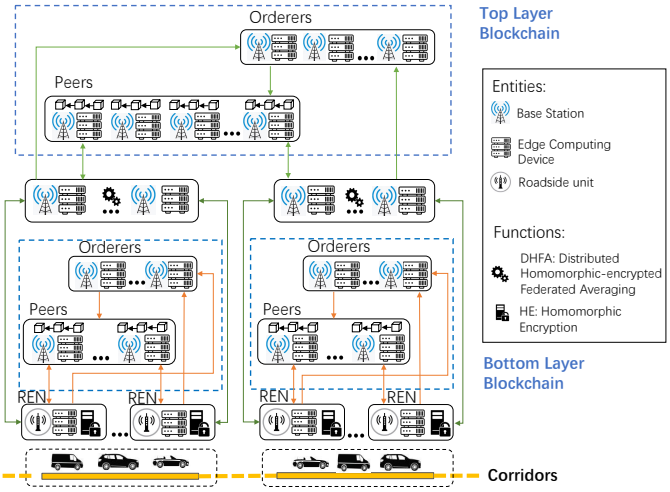


Fig. 1: A Bi-level Blockchained Federated Learning Architecture for Traffic Prediction.

et al. [22] proposed the privacy-preserving blockchain-based federated learning for traffic flow prediction task, they also applied a differential privacy method with a noise-adding mechanism to prevent data poisoning attacks. Qi et al. [23] proposed a blockchain-based, integrated homomorphic FL model to address the information silo problem. It provided gradient privacy protection by employing homomorphic encryption. Compared to our scheme, differential privacy methods need to add noise, and Qi's scheme [23] still utilizes one leader node to perform the cryptography task.

This research is the first effort to propose a bi-level blockchained architecture for secure federated learning-based traffic prediction. We designed the double-layer structure for blockchain storage and the DHFA algorithm to secure the federated averaging procedure.

III. System Architecture

This section describes a bi-level blockchained architecture for secure federated learning-based traffic prediction with intelligent vehicles. The overall architecture is divided into four components, where the corridors layer is based on regional information, including multiple intelligent vehicles and nearby REN. The bottom layer blockchain connected to the REN serves as the ledger to record the homomorphic encrypted local model parameters. The distributed homomorphic-encrypted federated averaging scheme serves as the generator for the global parameter aggregation procedure. Finally, the top layer blockchain acts as the ledger to record the global averaging model parameters results. The system structure is shown in Fig. 1. We first define the following entities:

- Base Station: A base station is a radio receiver/transmitter that serves as the hub of the local wireless network and can be the gateway between the REN, the wireless network, and other components.
- Edge Computing Device: Edge computing devices (ECs) are high-performance entities that can collaborate to accomplish the DHFA task.
- REN: A roadside edge node (REN) collects traffic data from the static sensing area along the road. It also serves as the information source for vehicles to collect traffic information.
- Bottom Layer Blockchain: The bottom layer blockchain serves as the ledger for regional sensors and manages the local model parameter within blockchain transactions. It forms a dynamic network based on the intelligent vehicles and REN geographic information.
- Top Layer Blockchain: Top layer blockchain is the coordinator for the global vehicular network. It stores global federated averaging parameter and other events data.
- Peer Node: A peer node is a blockchain entity that conducts transaction execution, endorsement, and block validation.
- Orderer Node: An orderer node is a blockchain entity that executes consensus procedures, which orders the transactions and batches them into new blocks, ensuring consistency and fault tolerance for the blockchain network.

In our proposed architecture, each REN, peer node, and orderer node includes a base station for communications and an edge computing device for computing and storage. Next, we define the following functions:

- Federated Learning (FL): Federated learning as a distributed machine learning approach enables nodes to collaboratively learn a shared prediction model. In our scenario, REN store and process the static sensor data to implement a federated learning process and train the local model parameter.
- Homomorphic Encryption (HE): Homomorphic encryption allows users to perform computations on encrypted data without decrypting it. We utilize homomorphic encryption to protect the privacy of the training model parameter for outsourced computation and storage tasks.
- Distributed Homomorphic-encrypted Federated Averaging (DHFA): DHFA is the algorithm that updates the global model parameter by calculating the average of the HE-encrypted local model parameters received from the clients. We utilize the partial homomorphic encryption scheme to protect the privacy of the model parameters.

As shown in Fig. 1, the proposed framework consists of one Top Layer (TL) blockchain and multiple Bottom Layers (BLs) blockchain, in which different BL blockchains are responsible for recording regional vehicular and local model parameters. Vehicles can communicate with each other and REN. Vehicles follow diverse roadway routes and collect data in different corridors. Each REN operates as a worker in the federated learning process. The HE-encrypted local model parameters are stored in the bottom layer blockchain transactions.

Additionally, the edge computing devices in the DHFA groups communicate with the REN and their edge computing devices at the corridors level. Among the N edge computing devices, $N-1$ edge computing devices and one randomly chosen edge computing device execute the DHFA algorithm cooperatively. Later, the global averaging parameters are collected by the peers and orders node, then recorded in top layer

Authorized licensed use limited to: Xi'an Jiaotong-Liverpool University. Downloaded on October 05,2023 at 08:02:57 UTC from IEEE Xplore. Restrictions apply.
© 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Services Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TSC.2023.3318990

blockchain transactions. The top layer blockchain stores the global federated averaging parameters, which can be used for traffic analysis in the future.

A. Problem Statement

A typical federated learning process encompasses the federated averaging algorithm of McMahan et al. [24], and a server (for instance, a service provider) orchestrates the training process by repeating the following steps until the training is stopped:

1) Client selection: The server samples from clients meeting eligibility requirements. For instance, mobile phones can check in to the server if they are plugged in, and idle to avoid impacting the user of the device.
2) Broadcast: The data users download the current model weights and the training program (e.g., a traffic model prediction) from the server.
3) Client computation: Each selected device locally computes an update to the model by executing the federated training program, which runs the SGD (Stochastic Gradient Descent) on local data.
4) Aggregation: The server collects an aggregate of the device updates. This stage can be the integration point for many other techniques, including secure aggregation for added privacy, and noise addition and update clipping for differential privacy.
5) Model update: The server updates the shared model based on the aggregated update computed from the clients who participated in current round [7].

A typical assumption in a federated learning system is that the participants are honest, whereas the server is honest-but-curious (HBC). As shown in Step 4 and Algorithm 1, with the client updates (i, x_t) as the input, line 1 shows the system initialization procedure, and from lines 2-4, for each round of the training process, it gets the random set of M clients. Line 5-7 indicates that for each client i in parallel, it calculates the $\sum_{k=1}^{M} \frac{1}{M} x^i_{1+t}$ as the x_{t+1} federated averaging result. As we can observe from Algorithm 1, the server collects an aggregate of the device updates, which may leak sensitive information.

Security Definition (HBC Adversary): A federated learning system typically assumes honest participants and security against an honest-but-curious server. That is, only the server can compromise the privacy of participants' data. Therefore, no information leakage from any participants to the server is allowed [25].

To address the above security issue, we propose the **secure computation problems of interest**. Secure aggregation is functionality for n clients and a server. It enables each client to submit a value, such that the server learns an aggregate function of clients' values, typically the sum value [7]. We integrated the homomorphic encryption scheme into the federated averaging procedure to secure the FL training process.

In this study, we propose the DHFA algorithm to conduct the global parameter averaging process based on the inputs of the local model. We improved the traditional federated learning algorithm and enhanced the privacy-preserving concerns when performing the federated averaging procedure. Nodes on the blockchain are responsible for recording both local model parameters and global averaging parameters, and the distributed federated learning parameter aggregation procedure is realized.

Algorithm 1: The server executes the federated averaging process.

Input : Client updates (i, x_t)
Output: Federated averaging x_{t+1}
1 initialize x_0;
2 **for** *each round* $t = 1, 2, ..., T$ **do**
3 $\quad S_t \longleftarrow random\ set\ of\ M\ clients$;
4 **end**
5 **for** *each each client* $i \in S_t$ *in parallel* **do**
6 $\quad x_{t+1} \longleftarrow \sum_{k=1}^{M} \frac{1}{M} x^i_{1+t}$
7 **end**
8 ▷ Federated averaging, when all clients have the same amount of data.

B. Permissioned Blockchain Network

B²SFL adopts a permissioned blockchain network (Hyperledger Fabric) as the FL framework, which includes the entities of peer nodes and orderer nodes defined as follows [26], [27]. Peers nodes P represent the entities that conduct transaction execution, endorsement, and block validation in both layers. Orderers nodes O are the nodes that execute consensus procedures that order the transactions and batch them into new blocks, ensuring consistency and fault tolerance for the blockchain network. In our architecture, REN and DHFA groups are the clients C who interact with the bottom and top layer blockchains by sending transaction requests. In particular, we adopt Hyperledger Fabric as the platform to emulate the proposed permissioned blockchain in this paper because it offers a rich open-source community, modular design, deployment flexibility, and high parallelization capabilities due to its execute-order-validate transaction workflow [28].

As shown in Fig. 1, we first map each component to the permissioned blockchain architecture based on Hyperledger Fabric [29]. RENs, consisting of a static sensor paired with an edge computing device, collect the traffic data at a fixed position within each corridor region. Later, each REN within a regional group will participate in the local model training process and encrypt their local model parameters. Then, acting as clients, each REN will interact with peers and orderers nodes within the bottom layer blockchain. After one round of the FL process, each REN submits a transaction proposal containing the HE-encrypted local model parameter to all the edge computing devices acting as peers for endorsements [26]. Each peer evaluates the transaction proposal and sends back an endorsement to the client (REN). Each REN packages the received endorsements into a transaction and submits it to the edge computing devices, which act as orderer nodes. The orderer nodes will execute the consensus algorithm to establish an exact order on blockchain transactions and batch them into

Authorized licensed use limited to: Xi'an Jiaotong-Liverpool University. Downloaded on October 05,2023 at 08:02:57 UTC from IEEE Xplore. Restrictions apply.
© 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Services Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TSC.2023.3318990

a new block [26]. Consequently, the local model parameters of each participating edge device are stored on the respective regional BL BC.

After sufficient local models are deposited to a given regional BL BC, the DHFA group will act as the client to compute the homomorphic-encrypted federated averaging results for the global parameters and subsequently store them in the top layer blockchain. The DHFA group will interact with the peers and the orderer nodes in the top-layer blockchain. After one round of the federated averaging process, the DHFA group will submit a transaction proposal containing the DHFA-protected global model parameter to all the edge computing devices acting as peers for endorsements. After the same evaluation process, the peer nodes will send back an endorsement to the DHFA group. The DHFA group packages all the received endorsements into a transaction and submits it to the top layer orderer nodes. The orderer nodes will establish an unambiguous order on blockchain transactions and batch them into a new block on the top layer blockchain.

For the blockchain ledger, the data structure used for the stored model updates is as follows:

```
type ModelUpdate struct {
    FederatedID         string
    LocationID          string
    DetectorID          string
    RoundNumber         int
    ModelParameters     HDF5
}
```

The parameters within the $ModelUpdate$ data structure represent the following: $FederatedID$ refers to the name of the associated FL process (e.g., "GRU TFP I-95"); $LocationID$ uniquely identifies the region associated with a given local model; $DetectorID$ is the identifier for a specific client $c \in \mathcal{C}$ within the blockchain network; $RoundNumber$ indicates the FL round index (i.e. i in r_i) associated with a given HDF5 file; and lastly the $ModelParameters$ field contains the HDF5 file.

C. Online Federated Learning of Traffic Prediction

B²SFL trains an online traffic flow prediction model at the network edge using data collected in real-time by traffic sensors without exchanging the collected data, resulting in dynamic and efficient-to-update models. In the proposed architecture, the clients $\mathcal{C}$ within a given geographic region e_i collaboratively train a single, regional prediction model G_e using Algorithm 4, leveraging the new incoming traffic data in a series of communication rounds $R = \langle r_1, r_2, \cdots, r_i, \cdots \rangle$.

In our system, the FL process is synchronous and the RENs update the global model synchronously in each FL round. During each round r, every client $c \in \mathcal{C}$ performs the following sequence of operations: (1) collect local traffic data; (2) train their local copy of the regional model with the collected data; (3) encrypt the local model parameters using HE-IBE and DHFA; (4) Encapsulate the local model parameters in a blockchain transaction and send it to the regional BL blockchain peers for endorsement and validation; (5) after collecting the necessary endorsements, package them into an endorsed transaction and send it to the regional BL blockchain orderers; (6) wait to receive the updated regional parameters from the DHFA group; (7) decrypt the updated regional parameters; and lastly, update their local version of the regional prediction model G_e.

Algorithm 2: Operations of clients $\mathcal{C}$ in region e_i during round r_i

1 **for** *each client* $c \in \mathcal{C}$ *in region* e_i *of* E *during round* r_i *of* R *in parallel* **do**
2 $\quad d_c^{new,i} \leftarrow c$.UPDDATASET($d_c^{old,i}$) ▷ Algo. 3;
3 $\quad$ ▷ $d_c^{new,i}$ becomes $d_c^{old,i+1}$ in r_{i+1}
4 $\quad \vec{p_c}^{\,i} \leftarrow c$.TRAINLOCALMODEL($\vec{p_c}^{\,i-1}$, $d_c^{new,i}$);
5 $\quad E_{pk}(\vec{p_c}^{\,i}) \leftarrow c$.ENCRYPT($\vec{p_c}^{\,i}$);
6 $\quad tx_c^i \leftarrow c$.TRANSACTIONPROPOSAL($E_{pk}(\vec{p_i})$);
7 $\quad c$.SENDTOPEERS(tx_c^i, $\mathcal{P}$);
8 $\quad etx_c^i \leftarrow c$.GETENDORSEDTRANSACTION($\mathcal{P}$);
9 $\quad c$.SENDTOORDERERS(etx_c^i, , $signature_c$, $\mathcal{O}$);
10 $\quad E_{pk}(\vec{p_e}^{\,i}) \leftarrow c$.RECEIVEUPDATEDPARAMS();
11 $\quad \vec{p_e}^{\,i} \leftarrow c$.DECRYPT($E_{pk}(\vec{p_e}^{\,i})$);
12 $\quad G_e^i \leftarrow c$.UPDATEMODEL($\vec{p_e}^{\,i}$);
13 **end**
14 Begin round r_{i+1};

Algorithm 3: Online traffic data collection and training data update of client c in round r_i

Input : $d_c^{old,i}$
Output: $d_c^{new,i}$
1 **while** *within the data collection period* p **do**
2 $\quad d_c^{in,i} \leftarrow c$.COLLECTDATA();
3 $\quad d_c^{new,i} \leftarrow d_c^{old,i} \cup d_c^{in,i}$;
4 **end**
5 **if** $d_c^{new,i}.size > MaxDataSize$ **then**
6 $\quad RemoveSize \leftarrow d_c^{new,i}.size - MaxDataSize$;
7 $\quad d_c^{new,i}$.REMOVEOLDDATA($RemoveSize$);
8 **end**
9 **return** $d_c^{new,i}$

During initialization, each client $c \in \mathcal{C}$ within a given region is provided an identical model G_e^0 before the first round r_1. G^0 could optionally be a pre-trained model to jump-start the learning process. At the start of round r_i, each client $c \in \mathcal{C}$ collects the incoming local traffic data $d_c^{in,i}$ (Algo. 2: line 2) for a predefined period p to be combined with its local historical traffic data $d_c^{old,i}$ to form $d_c^{new,i}$ (Algo. 3: lines 3-6). To mitigate overfitting of the old data and account for storage limitations, data samples in $d_c^{new,i}$ should be limited to a maximum data sample size $MaxDataSize$, where sensing data collecteddata in less recentfrom the older rounds areis excluded from the online training batchdataset in the more recentnew rounds (Algo. 3: lines 6-7). In particular, $MaxDataSize$ only controls the temporal range of the sensing measurements used for local training. On the contrary, the entire history of local and global model updates for all FL processes is stored on the respective bottom-layer blockchain

Authorized licensed use limited to: Xi'an Jiaotong-Liverpool University. Downloaded on October 05,2023 at 08:02:57 UTC from IEEE Xplore. Restrictions apply.
© 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Services Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TSC.2023.3318990

to provide auditability and tamper resistance.

After the data collection period, all clients $c \in \mathcal{C}$ update their local copy of the regional prediction model G_e obtained during the last FL round G_e^{i-1} using $d_c^{new,i}$ (Algo. 2: line 4). Next, the updated local parameter vector $\vec{p_e}^i$ is encrypted with the regional key through HE-IBE and is encapsulated within a transaction proposal tx_c^i. The proposal is sent to the peers $\mathcal{P}$ of the regional BL blockchain, who verify the transaction details and return an endorsement upon successful verification. Once the necessary endorsements are received, they are packaged into an endorsed transaction etx_c^i including the client's c digital signature $signature_c$ and transmitted to the orderer nodes $\mathcal{O}$. After receiving the transactions containing local model vectors $\vec{p_e}^i$ from all clients $c \in \mathcal{C}$, the orderer nodes $\mathcal{O}$ reach agreement on their state and package them into a block for storage on the regional BL blockchain and send the new data block to the peers $\mathcal{P}$. Next, the clients $c \in \mathcal{C}$ wait for the DHFA group to finish the execution of Algo. 4 and subsequently receive the updated encrypted model vector $E_{pk}(\vec{p_e}^i)$. Lastly, the vector is decrypted, and the new parameters are used to update the local model copy of each regional client $c \in \mathcal{C}$, resulting in G_e^i.

D. Distributed Homomorphic-Enhanced Global Model Uploading Protocol

First, an identity-based encryption (IBE) scheme integrated with homomorphic encryption is proposed to address the centralized authority issue. The key authority center (KAC) and REN have secret coordinate information. They could use secret coordinates to calculate the slope of two points and one line to generate the edge computing device's partial private key. It reduces the communication interactions to create the user's partial private key, reducing the risk of disclosing sensitive information in the DHFA interaction process.

1) HE-IBE Partial Private Key Distribution Protocol: This scheme is operated by the key authority center KAC and the REN, jointly managing and distributing edge computing devices' partial private keys. KAC is responsible for $EC's$ identity authentication and authorization, distributing the certification for ECs with a unique identifier, and generating the identity-based private key. KAC and REN utilized homomorphic encryption algorithms to generate the $EC's$ partial private key. The ECs communicate with KAC and REN to get the partial private key. HE-IBE ensures that the KAC and REN are not known to each other so that a curious third party can not decrypt $EC's$ partial private key.

Suppose KAC and REN have their confidential coordinate (x_{KAC}, y_{KAC}) and (x_{REN}, y_{REN}). We randomly pick (r_{xKAC}, r_{yKAC}) and then encrypts the coordinate with its public key:

$c_{xKAC} = (1+n)^{xKAC}\ r_{xKAC}^n\ mod\ n^2$;

$c_{yKAC} = (1+n)^{yKAC}\ r_{yKAC}^n\ mod\ n^2$;

then send c_{xKAC}, c_{yKAC} to the REN.

Similarly, when REN receives the c_{xKAC}, c_{yKAC}, it will randomly pick the k_{x1}, k_{y1}, where $k_{x1} \neq k_{y1}$. REN encrypts its coordinate information:

$c_{xREN} = (1+k_{x1}n)^{-xREN}\ r_{xREN}^n\ mod\ n^2$;

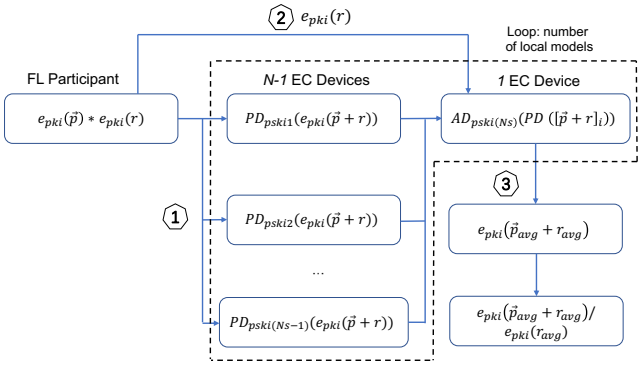


Fig. 2: Distributed Homomorphic-encrypted Federated Averaging Model.

$c_{yREN} = (1+k_{y1}n)^{-yREN}\ r_{yREN}^n\ mod\ n^2$;

Finally, KAC and REN compute the slope according to their private coordinate information.

The $EC's$ partial private key generation processes are shown as below:

Setup$(1^\lambda) \longrightarrow SysPara$: System initialization, it generates the system global parameter $SysPara$.

IKeyGen() $\longrightarrow PK_{KAC}, MK_{KAC}$: KAC generates public and private key pairs PK_{KAC}, MK_{KAC}.

RENKeyGen() $\longrightarrow PK_{REN}, MK_{REN}$: REN generates public and private key pairs PK_{REN}, MK_{REN}.

$KeyP_{KAC}(X_{KAC}, Y_{KAC}) \longleftrightarrow KeyP_{REN}(X_{REN}, Y_{REN})$: Both KAC and REN generate their coordinate (X_{KAC}, Y_{KAC}) and (X_{REN}, Y_{REN}), which include the public key information. Then KAC obtains the $EC's$ partial private key $SK_{ec(kac)}$, and REN interacts with KAC to get partial $SK_{ec(ren)}$ and sent it to ECs.

In the last, the generated partial private keys $SK_{ec(kac)}$ and $SK_{ec(ren)}$ will be sent to edge computing devices for the DHFA scheme constructions.

2) Distributed Homomorphic-encrypted Federated Averaging Scheme: As shown in Fig. 2, we propose a distributed homomorphic-encrypted federated averaging (DHFA) scheme which updates global model parameter by calculating the average of HE-encrypted local model parameters received from clients. Clients and edge computing devices ECs utilize the partial homomorphic encryption scheme to obtain the global model averaging parameters.

First, the client obtains the encrypted local model parameter value $E_{pk}(\vec{p_i})$ received from the REN, together with the encrypted random value $E_{pk}(r_i)$ utilizing homomorphic multiplication scheme, and the r is the random integer value generated by the client $c \in \mathcal{C}$. Then the client forwards the $E_{pk}(\vec{p_i}) \times E_{pk}(r_i)$ to $N-1$ edge computing devices (ECs). The $(N-1)$ ECs will perform partially homomorphic decryption operations, this process will be repeated for multiple times with all local parameters. Next, one random EC executes the additive decryption operation to get summation results of the local parameter value with random number $(m+r)$. The client will send the $E_{pk}(r_i)$ value to that random EC. One random EC calculates the average of local parameter value

Authorized licensed use limited to: Xi'an Jiaotong-Liverpool University. Downloaded on October 05,2023 at 08:02:57 UTC from IEEE Xplore. Restrictions apply.
© 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Services Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TSC.2023.3318990

Algorithm 4: Distributed Homomorphic-encrypted Federated Averaging Scheme

Input : Local model vectors $[\vec{pl}]_1, [\vec{pl}]_2, ..., [\vec{pl}]_{N_c}$
Output: Aggregated Global model vectors for clients $[\vec{pg}]_1, [\vec{pg}]_2, ..., [\vec{pg}]_{N_c}$

1 Client generates N_c random values $R_1, R_2, ..., R_{N_c}$ and encrypted them with public keys;
2 **for** $i \leq N_c$ **do**
3 | Client gets $[\vec{HA}_i]_i \longleftarrow [\vec{pl}]_i \cdot [R_i]_i$;
4 **end**
5 Client sends $[[\vec{pl}]_1 \cdot [R_1]_1, [\vec{pl}]_2 \cdot [R_2]_2, ..., [\vec{pl}]_{N_c} \cdot [R_{N_c}]_{N_c}$ to the $N-1$ ECs;
6 **for** $i \leq M$ **do**
7 | $N-1$ ECs partially decrypts the $PD[\vec{HA}_i]_i$ using sk_{ec} and obtains $(\vec{pl}_1 + R_1), ..., \vec{pl}_{N_c} + R_{N_c})$;
8 | One random EC generates the $\vec{p}_{sum_i} = \frac{\sum_k \vec{pl}_k + \sum_k R_k}{N_c}$;
9 **end**
10 **for** $i \leq N_c$ **do**
11 | ECs encrypts $\vec{p}_{sum_i}$ utilizing the public key pk_i;
12 **end**
13 One random EC gets the encrypted value $[[\vec{p}_{sum}]_1, [\vec{p}_{sum}]_2, ..., [\vec{p}_{sum}]_{N_c}]$;
14 Client sends the $E_{pk}(r_i)$ to one random EC device;
15 **for** $i \leq N_c$ **do**
16 | EC gets $[\vec{pg}]_i \longleftarrow [\vec{p}_{sum}]_i \div [\frac{\sum_k^m (R_k)}{N_c}]_i$;
17 **end**

with random number $(\vec{p_{avg}} + r_{avg})$, and the client sends the $E_{pk}(r_i)$ to that random EC device.

Finally, that random EC removes the averaged random values $E_{pk}(r_{avg})$ from the summation average value through the reversed homomorphic addition operation, and gets encrypted average local value $E_{pk}(\vec{p_{avg}})$ since the random integer value is generated by the client. Lastly, the ECs send the encrypted $(\vec{p_{avg}})$ to clients. The clients could decrypt the $(\vec{p_{avg}})$ using the symmetric HE keys and conduct the operation for the next round of the federated training process.

Algorithm 4 described the distributed secure averaging global parameter generation process. The local model vectors $[\vec{pl}]_{N_c}$ are the inputs and the aggregated global model vectors for clients $[\vec{pg}]_{N_c}$ are outputs. The ECs have the partial private keys sk_{ec}. The detailed operation of Algorithm 4 is as follows:

1) Client receives the HE-encrypted local model vectors from the REN. Local model vector encrypted with HE public key of $i-th$ client is described as $[\vec{pl}]_i$. Next, Client generates N_c random values R_{N_c} together with local model vector and encrypts them utilizing public key, shown as line 1;
2) The client conducts the homomorphic multiplication operations with HE-encrypted local model parameters $[\vec{pl}]_i$ together with the encrypted random vectors $[R_i]_i$ in line 3 when $i \leq N_c$. The homomorphic multiplication result is represented as $[\vec{HA}_i]_i$ This process is repeated for N_c local model parameters. Client sends $[\vec{HA}_{N_c}]_{N_c}$ to the $N-1$ ECs, as showing in line 5;
3) The $N-1$ ECs partially decrypts the $PD[\vec{HA}_i]_i$ utilizing the partial private key sk_{ec} and gets $\vec{pl}_{N_c} + R_{N_c}$. This process will repeat for M times when all the local parameters have been processed.
4) Next, one randomly chosen EC will generate all the encrypted summation values and divides the sum by total number of parameters N_c to get the vector including the average parameter represented as $\vec{p}_{sum_i}$ in line 8;
5) One random EC encrypts $\vec{p}_{sum_i}$ using the homomorphic public keys pk_i in lines 9 and then gets the encrypted values $[[\vec{p}_{sum}]_1, [\vec{p}_{sum}]_2, ..., [\vec{p}_{sum}]_{N_c}]$ in line 13;
6) The client sends the $E_{pk}(r_i)$ to one random EC device in line 14;
7) In last step, the random EC gets the DHFA-encrypted global model parameters by conducting the reversed homomorphic multiplication operation to remove the random parameter $[\frac{\sum_k^m (R_k)}{N_c}]_i$ as shown in line 16.

After the client and ECs finish the global parameter averaging procedure by performing the proposed DHFA algorithm, the one random EC sends the updated global model parameters to the peer nodes in the top layer blockchain. When REN start the next round of the federated learning process and send newly trained local model parameter to clients. Clients and ECs will execute the DHFA algorithm to generate the global averaging parameters again, and ECs will send back the encrypted averaging parameter $(\vec{p_{avg}})$ to clients, and the clients can decrypt the $(\vec{p_{avg}})$ using the symmetric HE keys.

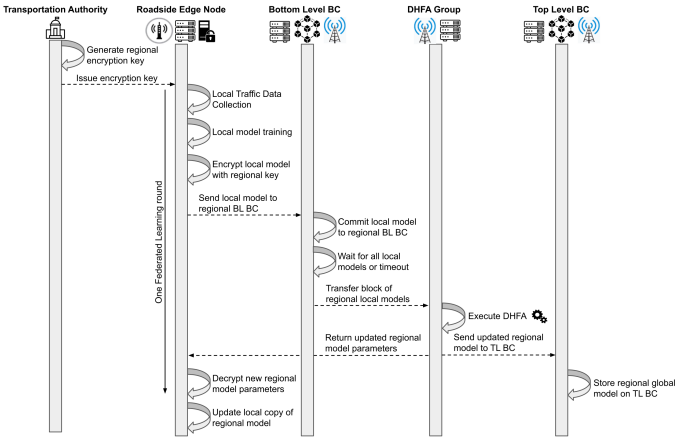


Fig. 3: Workflow of a bi-level blockchained architecture for secure federated learning-based traffic prediction.

E. Workflow of B²SFL

Figure 3 illustrates the federated learning workflow for online and location-aware traffic flow prediction within the proposed architecture. First, during initialization, the public/private key is generated for each region using the proposed HE-IBE (holomorphic encryption - identity-based encryption) and is distributed to the participants. Then, throughout the

Authorized licensed use limited to: Xi'an Jiaotong-Liverpool University. Downloaded on October 05,2023 at 08:02:57 UTC from IEEE Xplore. Restrictions apply.
© 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Services Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TSC.2023.3318990

FL process, the RENs within a pre-defined geographic region continuously collect local traffic data using their sensors (e.g., loop detector, LIDAR). During an FL round, after collecting sufficient local traffic data, each REN will leverage a combination of newly collected and historical data to train its local version of the global FL model. After training, the new model is encrypted using the regional key to protect the privacy and security of the local model parameters.

Next, REN encapsulates encrypted local model parameters within a blockchain transaction, which is sent to the peer nodes of their regional BL blockchain. The peer nodes validate the transaction and return a signed endorsement to REN. After receiving the necessary endorsements, the REN packages them into a digitally signed transaction and sends the transaction to the orderer nodes. After that, the orderer nodes will enact the consensus process, package newly submitted local models into a new block, and transfer the new block to the peers in their respective BL BC, where it is used to update the ledger state.

After block generation, the new block of updated regional global models is retrieved from the BL blockchain by the DHFA nodes acting as clients. To ensure fault tolerance of the synchronized FL process, we propose a timeout system parameter to account for communication or system outages for RENs. Specifically, the timeout value can be implemented at the DHFA node level using a client application, or implemented within the business logic of a respective BL blockchain. Once the local models of all participating RENs have been committed to the regional BL BC that manages the FL process, or when the current round timeout value is reached, the DHFA nodes will retrieve the local models and execute FedAVG.

After retrieving the local models, the proposed HE-FedAVG (Alg. 4) is executed to average the encrypted local model parameters into an updated version of the region-specific model. The newly encrypted regional model parameters are then transferred to the associated RENs. At the same time, the new parameters are also encapsulated in a blockchain transaction and are subsequently stored on the TL blockchain using the same transaction processing flow. Once the REN receives the updated model parameters, they are decrypted and used to update the REN's local copy of the regional prediction model, and the process repeats.

IV. System Analysis

We introduce and prove two propositions in this section to support the distributed homomorphic-enhanced global model uploading protocol to utilize the secret coordinates calculating the slope, and generating partial private keys without leaking any sensitive information.

Proposition 1. *The secure slope symbol calculation process can preserve the information leakage during the partial private key generation procedure.*

Proof. We define the

$$P(X) = \begin{cases} +1, X > 1 & \text{positive slope} \\ -1, X < 1 & \text{negative slope} \end{cases} \tag{1}$$

If $\frac{x}{y} > 1$ (x and y are point coordinates), then we have $\frac{x}{y} > \frac{x+m}{y+m} > 1$, else if $\frac{x}{y} < 1$, then we have $\frac{x}{y} < \frac{x+m}{y+m} < 1$. As a result, we get the following results ($m > 0$):

$$P(\frac{x_a}{x_b}) = P(\frac{x_a + m}{x_b + m}), \tag{2}$$

$$P(\frac{y_a}{y_b}) = P(\frac{y_a + m}{y_b + m}). \tag{3}$$

If we set:

$$P(X' - Y') = \begin{cases} +1, X' > Y' & \text{positive case} \\ -1, X' < Y' & \text{negative case} \end{cases} \tag{4}$$

Then we can indicate that:

$$P(\frac{x_a}{x_b}) = P'(x_a - x_b), \tag{5}$$

$$P(\frac{x_b}{x_a}) = P'(x_b - x_a), \tag{6}$$

and

$$P(\frac{y_a}{y_b}) = P'(y_a - y_b), \tag{7}$$

$$P(\frac{y_b}{y_a}) = P'(y_b - y_a). \tag{8}$$

As a result, partial private key distribution protocol can calculate the symbol of the slope correctly, and there is no information leakage during partial private key generation procedure. ■

Proposition 2. *The protocol of two private points secret equation of a line can correctly solve the equation of a line through two private points without leaking sensitive information.*

Proof.

$$Enc(A, (x_a, y_a)) = ((1+n)^{x_a} r_{x_a}^n mod n^2, \\ (1+n)^{y_a} r_{y_a}^n mod n^2)$$

$$Enc(B, (x_b, y_b)) = ((1+k_{x1}n)^{x_b} r_{x_b}^n mod n^2, \\ (1+k_{y1}n)^{y_b} r_{y_b}^n mod n^2)$$

Here k_{x1} and k_{y1} are two different random numbers.

$$Enc(A, (x_a^{k_{x1}}, y_a^{k_{y1}})) = ((1+n)^{k_{x1}x_a} r_{x_a}^{k_{x1}n} mod n^2, \\ (1+n)^{k_{y1}y_a} r_{y_a}^{k_{y1}n} mod n^2)$$

Next, B calculates the $Eny(k_{x1}\Delta x)$ and $Eny(k_{y1}\Delta y)$ through the (c_{x_a}, c_{y_a}):

we have

$$k_{x1}\Delta x = k_{x1}(x_a - x_b)$$

$$k_{y1}\Delta y = k_{y1}(y_a - y_b)$$

$$c_{k_{x1}\Delta x} = (1 + k_{x1}n)^{x_a - x_b}((r_{xb})^{n-x_b} r_{x_a}^{k_{x1}})^n mod\ n^2$$

Authorized licensed use limited to: Xi'an Jiaotong-Liverpool University. Downloaded on October 05,2023 at 08:02:57 UTC from IEEE Xplore. Restrictions apply.
© 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Services Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TSC.2023.3318990

$$c_{k_{y1}\Delta y} = (1+k_{y1}n)^{y_a-y_b}((r_{yb})^{n-y_b}r_{y_a}^{k_{y1}})^n mod\ n^2$$

After that, we get the $(c_{y1}, c_{x1}, (c_{y2}, c_{x2}, ..., (c_{yl}, c_l)$; and then A can compute the slope K:

$$\frac{L(c_{y1}^{\lambda} mod n^2)}{L(c_{x1}^{\lambda} mod n^2)} \times \frac{L(c_{y1}^{\lambda} mod n^2)}{L(c_{x1}^{\lambda} mod n^2)} \times \cdots \frac{L(c_{y1}^{\lambda} mod n^2)}{L(c_{x1}^{\lambda} mod n^2)} = \frac{L(c_{y1}^{\lambda} mod n^2) \cdot L(c_{y1}^{\lambda} mod n^2) \cdot L(c_{y1}^{\lambda} mod n^2)}{L(c_{x1}^{\lambda} mod n^2) \cdot L(c_{y1}^{\lambda} mod n^2) \cdot L(c_{y1}^{\lambda} mod n^2)} = \frac{\Delta y}{\Delta x}(k_{x1} \cdot k_{x2} \cdot ... \cdot k_{xl} = k_{y1} \cdot k_{y2} \cdot ... \cdot k_{yl}) = K$$

K is the slope of the line which goes through A and B, and A computes the line equation:

$$y = K(x - x_a) + y_a$$

Similarly, when B receives the K, it also computes the line equation:

$$y = K(x - x_b) + y_b$$

The information A sends to B is ciphertext during the partial private key construction process, and B cannot decrypt the information since it does not have the private key. Also, A cannot calculate the coordinate information of B. The secret information of either A or B was not leaked during the partial private key generation process, the slope of the linear equation was correctly solved, and the system correctness was proved. ■

V. Security Analysis

In traditional FL, failure of a participant or central server can negatively impact FL performance. In this section, we discuss threat model from two perspectives: server vulnerabilities and participant vulnerabilities.

A. Single Point of Failure Attack

The central server is a vital coordinator for FL, collecting and aggregating all local model updates. Suppose the central server suffers a typical single point-of-failure attack. In that case, the execution of the model update aggregation will fail, and the new global model will not be assigned to the participant for subsequent local model training, which means the entire FL algorithm terminates. Our proposed scheme can avoid this issue since the FL process is done by the DHFA, which has a distributed architecture. Compared to one centralized FL server, $(N-1)$ ECs will perform partially homomorphic decryption operations, this process will be repeated for multiple times with all local parameters.

B. Member Inference Attack

Since the central server knows local model updates from all participants, the central server can record the parameter of the local model. The central server can further utilize the parameter information of the local model to perform membership inference attacks to steal the local data sets of the participants. In this case, participants have no way of knowing whether their parameter information is being logged by the central server. Our proposed system can solve this problem since the parameter, and local models are encrypted with HE, which can not be directly exposed. HE allows data to be encrypted while performing computation on other computing devices.

C. Label Flipping Attack

During the execution of the FL process, malicious participants may modify the tags of the local data set to provide low-quality model updates, which will affect the entire federated learning phase. For instance, a vehicle can change traffic flow data at time t, affecting traffic flow prediction at that time $t+1$. However, such an attack is not possible in our system. All the local parameter and data is stored in the bottom blockchain as the permanent record. Also, once the encrypted model has been uploaded to the top-layer blockchain, which can not be modified again to change the label. Blockchain is immune to the label flipping attack.

VI. Experiments and Evaluations

A. Performance of Blockchain Network

1) Setup: The blockchain modular was developed with Hyperledger Fabric blockchain v2.2. It was deployed and experimented on the virtual machine with 3.70 GHz Intel i9-10900K processor and 24GB of memory. We instantiated four peers and five orderers using the Raft consensus algorithm. In the experiments, we used Hyperledger Caliper tool to measure the blockchain performance in two metrics: transaction throughput and transaction latency [30].

Transaction throughput quantifies the number of transactions per second (tps) which can be successfully processed by the blockchain network, while the latency indicates the average running time of transactions from initial construction by the clients until successfully committed to the ledger. In all the experiments, we analyze both metrics by increasing transaction send rates which indicate the number of input transactions per second by the blockchain clients. Notably, the number of clients for each bottom layer blockchain is usually less than seven due to the bi-level design.

2) Transaction Throughput: In Fig. 4, we illustrate the throughput of our blockchain network. We report the average transaction throughput over multiple testing cycles to quantify performance. It shows that the transaction throughput increases as the send rate increases to 400 tps. However, the performance levels off at send rates above 350 tps, and throughput remain relatively constant at 270 tps. This indicates that the maximum network throughput is about 270 tps.

Authorized licensed use limited to: Xi'an Jiaotong-Liverpool University. Downloaded on October 05,2023 at 08:02:57 UTC from IEEE Xplore. Restrictions apply.
© 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Services Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TSC.2023.3318990

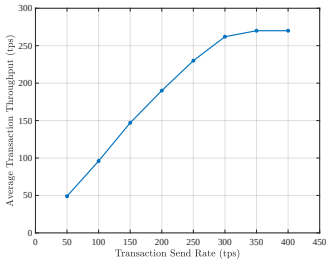


Fig. 4: Transaction throughput vs. transaction send rate.

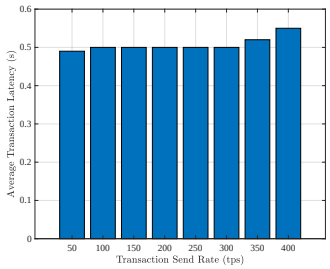


Fig. 5: Transaction latency vs. transaction send rate.

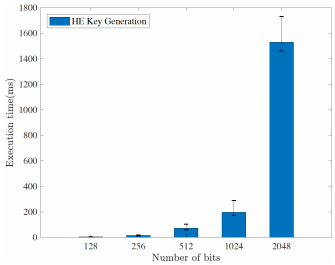


Fig. 6: Execution time of HE key generation process.

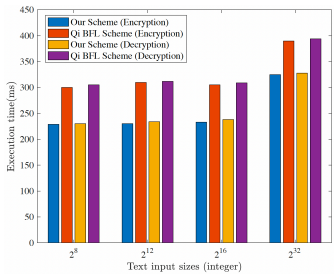


Fig. 7: Execution time comparison of our scheme, and Qi's scheme [23].

3) Transaction latency: In Fig. 5, we illustrate the latency of our blockchain network. The average transaction latency remains at 0.5 s when the transaction sends rate is below 300 tps. After passing 300 tps transaction send rates, the transaction latency increases slowly. The results indicate our blockchain network can handle the transaction requests from a certain number of RENs in a corridor. Compared to the centralized solution, our scheme is feasible in transaction latency for traffic management systems.

B. Homomorphic Encryption Execution Comparison Result

To evaluate distributed homomorphic-encrypted federated averaging cryptographic system performance, we tested the experiments based on jspaillier. We boosted the number of bits from 128 to 256, 512, 1024, and 2048 to measure the execution time of key pair generation procedures. As shown in Fig. 6, key generation time is 5 ms for 128 bits, 16 ms for 256 bits, 73 ms for 512 bits, 198 ms for 1024 bits, and 1531 bits for 2048 ms. It shows that key generation time will increase exponentially when the number of bits grows. As we can see from the result, a 1,024-bit key pair needs the 512-bit prime number for the key generation process, which provides sufficient security requirements.

Next, we compare the execution time of partial homomorphic addition operation ($(A + B)$), encrypted multiplication ($(A + B) * C$), and decryption ($(A + B) * C$) algorithms with Qi's scheme [23]. As we can see from Fig. 7, encrypted addition and decryption time for our scheme 2^{12} is 226 ms and 230 ms, which outperforms the Qi's scheme [23]. The encrypted addition and decryption time for 2^{32} are 307 ms and 325 ms in our scheme, and Qi's scheme [23] is around 400ms. Compared to other cases, 2^{32} case's execution cost all increases slightly. As a result, our proposed scheme could achieve better performance when increasing input data size for the traffic prediction task.

C. Federated Learning

1) Experimental Setup: The FL experiments are simulation-based and were conducted on Google Colab using one NVIDIA P100 GPU, two Intel(R) Xeon(R) CPUs @ 2.30GHz, and 13.34 gigabytes of RAM. All experiments trained a multi-layer GRU (gated recurrent unit) model for predicting traffic flow at a 5-minute horizon. In our GRU model design, we stack two GRU layers in sequence with varying numbers of hidden units, followed by a dropout layer with the final layer as a densely connected output layer. To simulate online training, the input samples were dynamically fed in sequence during each FL round. At the start of each experiment, the parameters of G^0 are randomly initialized without a pre-trained model. Moreover, in addition to the federated models, we train a baseline reference model for comparative analysis. The baseline model architecture is the same as the federated model, but it is trained without federation exclusively using the data of a given individual detector.

2) Dataset and Study Area: We use a real-world traffic flow dataset in our experiments. The Delaware Department of Transportation (DelDOT) provides the dataset and includes traffic flow data collected from DelDOT-maintained roadways at a 5-minute resolution. We select various nonsequential detectors along the I-95 north arterial to act as the FL clients in each experiment. Each client is provided a location-specific dataset containing point-based flow measurements from the start of August 2019 until the end of September 2019. We separate 80% of the data for real-time training and inference while saving the remaining 20% for future offline inference experiments.

3) Comparison with Other Blockchain-based Federated Learning Systems: In this subsection, we compare the different blockchain types, architecture, encryption methods, privacy protection, decentralized FedAvg, and the blockchain plat-

Authorized licensed use limited to: Xi'an Jiaotong-Liverpool University. Downloaded on October 05,2023 at 08:02:57 UTC from IEEE Xplore. Restrictions apply.
© 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Services Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TSC.2023.3318990

Algorithm 5: Online inference of client c in round $r_i, i > 1$

Input : G^{i-1}, B_c^{i-1}, $d_c^{old,i}$
Output: None
1 **for** *each client* $c \in \mathcal{C}$ *in region* e_i *of* E *during round* r_i *of* R *in parallel* **do**
2 $d_c^{in,i}$, $BASE_c^i$, $FED_c^i \leftarrow$ [], [], []; ▷ Empty arrays.
3 $j \leftarrow 0$;
4 $d_c^{pred,j} \leftarrow d_c^{old,i}[: input\_shape]$;
5 ▷ Extract $input\_shape$ number of the latest data.
6 **while** $j < input\_shape$ **do**
7 $BASE_c^i$.ADD(c.PREDICTBY(B_c^{i-1}, $d_c^{pred,j}$));
8 FED_c^i.ADD(c.PREDICTBY(G^{i-1}, $d_c^{pred,j}$));
9 $d_c^{in,i}$.ADD(c.COLLECTONEDATA());
10 $j \leftarrow j + 1$;
11 $d_c^{pred,j} \leftarrow d_c^{pred,j}$.POPLEFT() $\cup$ $d_c^{in,i}$;
12 **end**
13 $TRUE_c^i \leftarrow d_c^{in,i}$;
14 **end**

form among our proposed scheme and other state-of-the-art blockchain-based federated learning systems, including the Li BFL [16], Zhao BFL [21], Fabric FL [19], BAFL [15], Biscotti BFL [12], Hierarchical BFL [31], and Qi BFL [23].

As we observe from Table I, most of the proposed blockchain-based federated learning systems utilize the permissioned blockchain type, and the one-layer blockchain architecture is popular. Only Qi BFL [23], and our proposed system offer partial HE features, and our scheme is based on the DHFA, which supports the decentralized FedAvg method. The bi-level blockchain architecture with DHFA encryption can protect data security in a more efficient and end-to-end privacy-preserving way. We provide detailed comparative experiments with Qi BFL [23] in the HE performance evaluation section. For the Li BFL [16], Zhao BFL [21], Fabric FL [19], BAFL [15], Biscotti BFL [12], and Hierarchical BFL [31], we conduct the table comparison results since their evaluation is based on other cryptography techniques, FL simulation results, and other evaluation metrics. Most existing schemes utilize differential privacy requiring adding noise for the data. Our proposed DHFA scheme can solve the centralized FL server issue and perform calculations on the encrypted data.

4) Regional Group Size Performance Comparison Analysis: Motivated by our previous work in [26], we select the GRU model as the chosen DL architecture for the FL experiments. Each FL simulation consists of 1165 rounds, where the duration of each round is one hour. In our dataset, sensors collect flow data at a 5-minute resolution, corresponding to twelve new data instances per hour per sensor. The GRU models are designed to process input instances sequentially, and one hour of historical flow data is used to generate the flow prediction for the next five minutes. As shown in Algo. 5, the sample size of $d_c^{in,i}, i > 1$ equals the $input\_shape$ of the GRU models (i.e., 12), to control data flow according to the hourly rounds. The exception is that $d_c^i.size, i = 1$ is set to 24 (i.e., 2 times of $d_c^{in,i}.size$, where $i > 1$) in r_1 for all $c \in \mathcal{C}$, because the models need at least 13 data samples for training. We set $MaxDataSize = 24$ in the regional group size performance analysis, providing the models with two hours of historical traffic flow data for training during each FL round. During simulation, $MaxDataSize$ controls the sample size of historical data to use when training for the current round. Moreover, we set the number of training epochs for each round to five.

B²SFL proposes a regional grouping of detectors within transportation networks to perform localized FL, producing region-specific online traffic prediction models. To elucidate the impacts of varying regional group sizes, we conduct FL simulations using identical GRU models while varying the number of participating clients (and, consequently, the number of distinct time series being considered). Fig. 8 illustrates the partial online inference curves for three traffic sensors. We examine FL group sizes of 3, 7, and 12 detectors within a single region, together with comparison with the baseline case B_c of a single detector. In the baseline case, the model is trained identically to the FL models, but DHFA is not invoked and the model is trained locally on a given REN and never leaves the device. The plots compare the regional global model prediction output with the ground truth sensor data and the baseline case for the last 24 FL rounds, translating to a day of traffic flow data. The process for obtaining real-time inference values is shown in Algo. 5. Specifically, for any client $c \in \mathcal{C}$ in round $r_i \in R, i > 1$, we generate a temporary dataset $d_c^{pred,j}$ by extracting $input\_shape$ of the most recent data points from $d_c^{old,i}$ (Algo. 5: line 4). Notably, $input\_shape = p$, where p is the collection period for each FL round. We define $p = 12$ in this study, thus each c collects one hour of new traffic data before training again. During data collection, each newly collected instance is appended to $d_c^{pred,j}$, while popping the oldest data instance to maintain a vector length of $input\_shape$ elements. After appending a new value, c uses both B_c^{i-1} and G^{i-1} to perform online inference (Algo. 5: lines 7-11), updating $BASE_c^i$ and FED_c^i with the resulting prediction. After completing an $input\_shape$ number of prediction steps, the resulting $BASE_c^i$ and FED_c^i will contain the inference values from the corresponding models for $r_i, i > 1$. Lastly, $d_c^{in,i}$ is assigned to $TRUE_c^i$ (Algo. 5: line 13), which represents the *TRUE* curve(s) in Fig. 8.

Examining Fig. 8 indicates that the group size directly impacts the resulting global model prediction accuracy. For example, comparing the $N = 3$ and $N = 7$ groups, we can see that increasing the group size to seven improved the prediction performance for both sensors 19992 and 19912, with 19912 experiencing the most improvement. On the other hand, the global model prediction accuracy for sensor 19985 suffered when the group size was increased. Notably, despite the spatial closeness of the three sensors in the $N = 3$ group, as shown in Fig. 8, we can see that the time series distribution differs in trend and magnitude between the sensors. Consequently, the prediction accuracy degrades when these sensors are grouped in a small regional cluster due to the federated averaging process for merging the parameters.

To view the differences in accuracy at a more granular level, Table II provides the mean absolute error (MAE), mean

Authorized licensed use limited to: Xi'an Jiaotong-Liverpool University. Downloaded on October 05,2023 at 08:02:57 UTC from IEEE Xplore. Restrictions apply.
© 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Services Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TSC.2023.3318990

TABLE I: Comparisons with other blockchain-based federated learning systems.

Proposed Scheme	Blockchain Type	Architecture	Encryption method	Privacy Protection	Decentralized FedAvg	Blockchain Platform
Li BFL [16]	Permissionless	One-layer	Adding Noise	Differential Privacy	×	×
Zhao BFL [21]	×	One-layer	Adding Noise	Differential Privacy	×	×
Fabric FL [19]	Permissioned	One-layer	×	×	Yes	Hyperledger Fabric
BAFL [15]	×	One-layer	×	Entropy Weight	×	×
Biscotti BFL [12]	Permissioned	One-layer	Secure Aggregation	Differential Privacy	Yes	×
Hierarchical BFL [31]	×	Hierarchical	×	×	×	×
Qi BFL [23]	Permissioned	One-layer	Aggregation	Partial HE	×	Ethereum
Our Scheme	Permissioned	Bi-level	Secure FedAvg	DHFA	Yes	Hyperledger Fabric

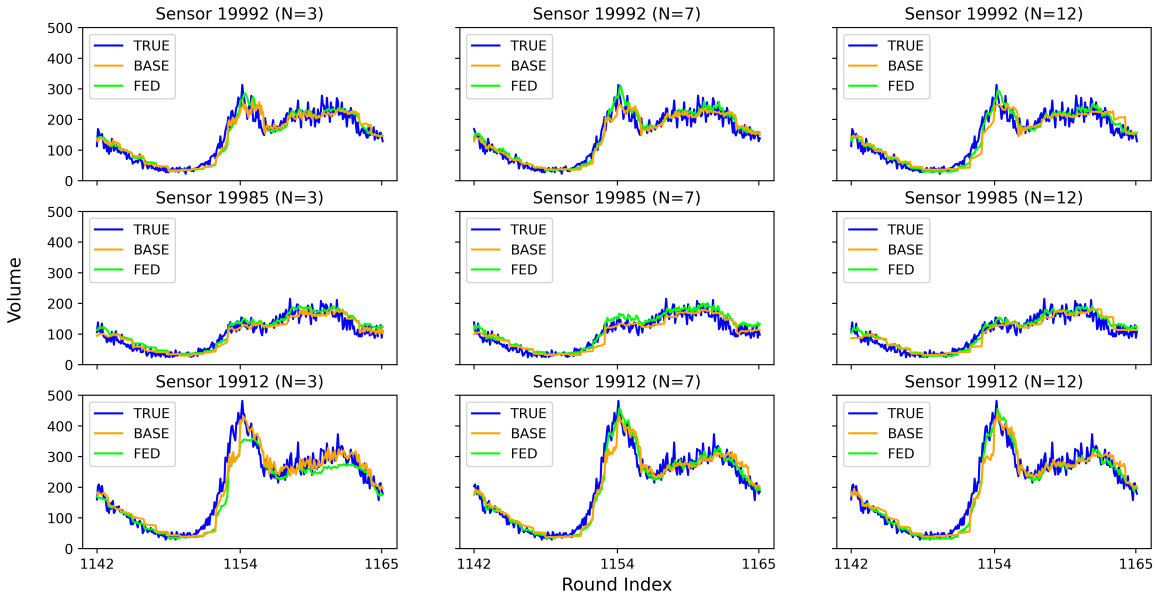


Fig. 8: Online prediction curves for the last 24 FL rounds with varying group sizes. The FL models are compared to the baseline cases for each reference detector.

squared error (MSE), root mean squared error (RMSE), and mean absolute percent error (MAPE) for three representative detectors present in the three experimental groups ($N = 3, 7, 12$), while the case $N = 1$ represents B_c, the baseline model without federation. Notably, we can see that a group size of seven resulted in the best performance for detectors 19912 and 19992, and the performance was significantly improved from the baseline and $N = 3$ cases. However, when further increasing the group size to 12, we see a decrease in performance across these two detectors. These results reveal that controlling for group size has an important impact on accuracy.

5) Reduced Parameter Model Performance Comparison Analysis: In this experiment, we analyze the online traffic prediction accuracy of a lightweight GRU model having considerably lower parameters than existing approaches [4]. While DHFA can be expensive to execute for deep models, recent research has highlighted that many of the learned parameters provide little impact on the prediction output [32]. Motivated by this observation, this experiment assesses the effects of drastically reducing the number of parameters in our FL-based online traffic flow prediction (TFP) model to reduce DHFA execution time.

TABLE II: Prediction error calculations under various metrics and group sizes (N)

Metric	Group Size	19912	19985	19992
MAE	N = 1	24.76	15.33	18.98
	N = 3	28.12	16.81	18.97
	N = 7	19.79	17.20	16.72
	N = 12	20.29	15.16	17.72
MSE	N = 1	1171.76	396.44	654.09
	N = 3	1666.78	434.75	621.95
	N = 7	717.44	479.97	489.48
	N = 12	753.69	360.59	512.59
RMSE	N = 1	34.22	19.89	25.47
	N = 3	40.83	20.85	24.94
	N = 7	26.79	21.91	22.12
	N = 12	27.45	18.99	22.64
MAPE	N = 1	0.15	0.18	0.16
	N = 3	0.15	0.21	0.16
	N = 7	0.12	0.20	0.14
	N = 12	0.13	0.18	0.15

When building the model, we instantiate seven participants using the same detectors and data set as the $N = 7$ group from Sec. VI-C4. The GRU model structure is the same, with two sequential GRU layers. However, each layer's hidden units is reduced to 5 (from 50 previously). We also set $MaxDataSize = 240$ and the number of epochs to fifty during training. Under this design, the number of trainable

Authorized licensed use limited to: Xi'an Jiaotong-Liverpool University. Downloaded on October 05,2023 at 08:02:57 UTC from IEEE Xplore. Restrictions apply.
© 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Services Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TSC.2023.3318990

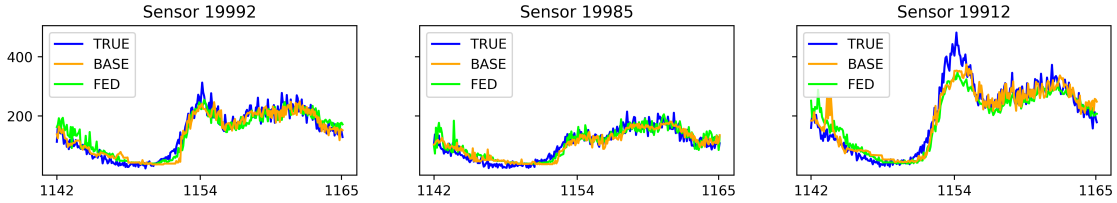


Fig. 9: Online prediction curves for the last 24 FL rounds using the reduced parameter model. The FL models are compared to the baseline cases for each reference detector.

TABLE III: Online inference errors during the last 24 FL rounds for the light-weight model

Detector	MAE	MSE	RMSE	MAPE
19912	31.28	1745.70	41.78	0.19
19924	81.13	11199.60	105.83	0.24
19951	51.51	5036.58	70.97	0.19
19978	27.24	1053.15	32.45	0.56
19985	18.02	548.59	23.43	0.23
19992	21.77	805.77	28.39	0.20
19997	34.30	2183.80	46.73	0.18

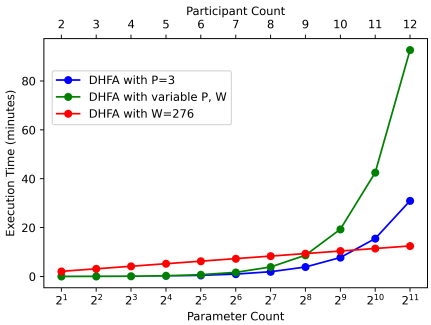


Fig. 10: Execution time complexity model for the proposed DHFA algorithm.

model parameters is reduced from 23,001 to 276. The FL simulation is conducted in the same manner mentioned in Sec. VI-C4. Fig. 9 illustrates the real-time comparative prediction curve for detectors 19992, 19985, and 19912, while Table III provides the prediction errors for each detector during the last 24 simulated communication rounds.

Using detector 19992 as a comparative reference, we can see that the MAPE prediction error increased from 0.14 to 0.20 when reducing the number of learnable parameters, representing a 42.86% change in accuracy. While this increase is substantial, it is notable that the reduction in learnable parameters between the two models represents a 98.8% decrease in weights.

6) Execution Time Complexity Model: This subsection presents the execution time complexity model for the proposed DHFA within our online traffic flow prediction workflow using GRU. Notably, DHFA performs two fundamental operations: encrypted addition and encrypted multiplication. During execution, the parameters of each participant in a given regional FL group will be securely averaged together using both operations. Consequently, the number of operations performed during each execution is a function of two parameters: W, representing the model's total number of learnable weights, and P, denoting the number of participants in a given regional FL group. Computing W is inherently specific to the model architecture, and in this analysis, we focus on the GRU model as an example. The GRU model is a recurrent neural network (RNN) consisting of three feedforward neural networks (FFNN) structured as a series of gates. The following equation can be used to calculate the number of trainable weights within a single FFNN:

$$W_i = (h(h+i)+h)$$

where h indicates the size of the hidden layer and i denotes the length of the input vector. Similarly, because GRU consists of 3 FFNNs, we can compute the total number of learnable parameters within a single GRU layer with the equation:

$$W = \sum_{n=1}^{3} W_i$$

For the first layer, $i_1 = 1$, because the traffic sequence is fed into the model sequentially. In the subsequent GRU layer, $i_2 = 5$ represents the number of hidden units in the first layer, which is five in our lightweight model. Our output layer in all models is a fully connected layer, where $W_i = i_2 + 1$ accounts for the hidden representation for i_2 neurons and the final output weight. Consequently, the total parameter count can be computed for our lightweight model using the presented equations to determine that $W = 276$.

For modeling the execution time of DHFA, we use the experimental values presented in Sec. VI-B for an input data size of 2^{12}: 226ms for encrypted addition and 230ms for encrypted multiplication. During execution, DHFA will compute $P-1$ additions and one multiplication for each of the W parameters in the model. Accordingly, the total execution time T can be estimated using the following equation:

$$T = W(226 * (P-1) + 230)$$

The results for various values of W and P are illustrated in Fig. 10. Within the figure, three calculations are presented: (1) the blue line computes T with a static $P = 3$; (2) the green line computers T with variable P and W at each point; and (3) the red line computes T with a static $W = 276$ corresponding to the value of W for our lightweight model. Notably, the bottom x-axis is scaled exponentially. In comparison, the unencrypted federated averaging algorithm generally runs on the order of

Authorized licensed use limited to: Xi'an Jiaotong-Liverpool University. Downloaded on October 05,2023 at 08:02:57 UTC from IEEE Xplore. Restrictions apply.
© 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Services Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TSC.2023.3318990

milliseconds due to its simplicity. Consequently, integrating DHFA is computationally expensive compared to a system without encryption and is suitable for privacy-centric ITS applications.

VII. Conclusion

This paper proposed a bi-level blockchain architecture for secure federated learning-based traffic prediction. The bottom and top layer blockchains store local and aggregated global parameters. We design the partial private key distribution protocol and the partially homomorphic encryption scheme to achieve the privacy-preserving federated averaging procedure. We conducted both system correctness and security discussions to validate our design. We implemented the proposed architecture by utilizing Hyperledger Fabric, Jspaillier library, and the Google Colab platform. The experiment results indicate that the proposed scheme is secure and efficient for decentralized federated averaging schemes and real-world traffic prediction tasks.

References

[1] K. Lee, M. Eo, E. Jung, Y. Yoon, and W. Rhee, "Short-term traffic prediction with deep neural networks: A survey," *IEEE Access*, vol. 9, pp. 54 739–54 756, 2021.

[2] J. Xu, B. S. Glicksberg, C. Su, P. Walker, J. Bian, and F. Wang, "Federated learning for healthcare informatics," *Journal of Healthcare Informatics Research*, vol. 5, no. 1, pp. 1–19, 2021.

[3] L. U. Khan, W. Saad, Z. Han, E. Hossain, and C. S. Hong, "Federated learning for internet of things: Recent advances, taxonomy, and open challenges," *IEEE Communications Surveys & Tutorials*, 2021.

[4] Y. Liu, J. James, J. Kang, D. Niyato, and S. Zhang, "Privacy-preserving traffic flow prediction: A federated learning approach," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 7751–7763, 2020.

[5] T. Zeng, J. Guo, K. J. Kim, K. Parsons, P. Orlik, S. Di Cairano, and W. Saad, "Multi-task federated learning for traffic prediction and its application to route planning," in *2021 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2021, pp. 451–457.

[6] M. Shaygan, C. Meese, W. Li, X. G. Zhao, and M. Nejad, "Traffic prediction using artificial intelligence: Review of recent advances and emerging opportunities," *Transportation Research Part C: Emerging Technologies*, vol. 145, p. 103921, 2022.

[7] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings *et al.*, "Advances and open problems in federated learning," *Foundations and Trends® in Machine Learning*, vol. 14, no. 1–2, pp. 1–210, 2021.

[8] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proceedings of the forty-first annual ACM symposium on Theory of computing*, 2009, pp. 169–178.

[9] H. Guo, W. Li, M. Nejad, and C.-C. Shen, "A hybrid blockchain-edge architecture for electronic health record management with attribute-based cryptographic mechanisms," *IEEE Transactions on Network and Service Management*, pp. 1–1, 2022.

[10] H. Chai, S. Leng, Y. Chen, and K. Zhang, "A hierarchical blockchain-enabled federated learning algorithm for knowledge sharing in internet of vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 7, pp. 3975–3986, 2021.

[11] B. Jia, X. Zhang, J. Liu, Y. Zhang, K. Huang, and Y. Liang, "Blockchain-enabled federated learning data protection aggregation scheme with differential privacy and homomorphic encryption in iiot," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 6, pp. 4049–4058, 2022.

[12] M. Shayan, C. Fung, C. J. Yoon, and I. Beschastnikh, "Biscotti: A blockchain system for private and secure federated learning," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 7, pp. 1513–1525, 2020.

[13] Y. Lu, X. Huang, K. Zhang, S. Maharjan, and Y. Zhang, "Blockchain empowered asynchronous federated learning for secure data sharing in internet of vehicles," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 4, pp. 4298–4311, 2020.

[14] Z. Peng, J. Xu, X. Chu, S. Gao, Y. Yao, R. Gu, and Y. Tang, "Vfchain: Enabling verifiable and auditable federated learning via blockchain systems," *IEEE Transactions on Network Science and Engineering*, vol. 9, no. 1, pp. 173–186, 2022.

[15] L. Feng, Y. Zhao, S. Guo, X. Qiu, W. Li, and P. Yu, "Bafl: A blockchain-based asynchronous federated learning framework," *IEEE Transactions on Computers*, vol. 71, no. 5, pp. 1092–1103, 2022.

[16] J. Li, Y. Shao, K. Wei, M. Ding, C. Ma, L. Shi, Z. Han, and H. V. Poor, "Blockchain assisted decentralized federated learning (blade-fl): Performance analysis and resource allocation," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 10, pp. 2401–2415, 2022.

[17] Y. Qu, S. R. Pokhrel, S. Garg, L. Gao, and Y. Xiang, "A blockchained federated learning framework for cognitive computing in industry 4.0 networks," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 4, pp. 2964–2973, 2021.

[18] J. Qi, F. Lin, Z. Chen, C. Tang, R. Jia, and M. Li, "High-quality model aggregation for blockchain-based federated learning via reputation-motivated task participation," *IEEE Internet of Things Journal*, pp. 1–1, 2022.

[19] V. Mothukuri, R. M. Parizi, S. Pouriyeh, A. Dehghantanha, and K.-K. R. Choo, "Fabricfl: Blockchain-in-the-loop federated learning for trusted decentralized systems," *IEEE Systems Journal*, pp. 1–12, 2021.

[20] H. Liu, S. Zhang, P. Zhang, X. Zhou, X. Shao, G. Pu, and Y. Zhang, "Blockchain and federated learning for collaborative intrusion detection in vehicular edge computing," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 6, pp. 6073–6084, 2021.

[21] Y. Zhao, J. Zhao, L. Jiang, R. Tan, D. Niyato, Z. Li, L. Lyu, and Y. Liu, "Privacy-preserving blockchain-based federated learning for iot devices," *IEEE Internet of Things Journal*, vol. 8, no. 3, pp. 1817–1829, 2020.

[22] Y. Qi, M. S. Hossain, J. Nie, and X. Li, "Privacy-preserving blockchain-based federated learning for traffic flow prediction," *Future Generation Computer Systems*, vol. 117, pp. 328–337, 2021.

[23] M. Qi, Z. Wang, F. Wu, R. Hanson, S. Chen, Y. Xiang, and L. Zhu, "A blockchain-enabled federated learning model for privacy preservation: System design," in *Australasian Conference on Information Security and Privacy*. Springer, 2021, pp. 473–489.

[24] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*. PMLR, 2017, pp. 1273–1282.

[25] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 2, jan 2019. [Online]. Available: https://doi.org/10.1145/3298981

[26] C. Meese, H. Chen, S. A. Asif, W. Li, C.-C. Shen, and M. Nejad, "Bfrt: Blockchained federated learning for real-time traffic flow prediction," in *2022 22nd IEEE International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*. IEEE, 2022, pp. 317–326.

[27] H. Guo, W. Li, and M. Nejad, "A location-based and hierarchical framework for fast consensus in blockchain networks," in *2021 4th International Conference on Hot Information-Centric Networking (HotICN)*. IEEE, 2021, pp. 1–6.

[28] J. Dreyer, M. Fischer, and R. Tönjes, "Performance analysis of hyperledger fabric 2.0 blockchain platform," in *Proceedings of the Workshop on Cloud Continuum Services for Smart IoT Systems*, ser. CCIoT '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 32–38. [Online]. Available: https://doi.org/10.1145/3417310.3431398

[29] W. Li, C. Meese, H. Guo, and M. Nejad, "Aggregated zero-knowledge proof and blockchain-empowered authentication for autonomous truck platooning," *IEEE Transactions on Intelligent Transportation Systems*, 2023.

[30] H. Guo, W. Li, and M. Nejad, "A hierarchical and location-aware consensus protocol for iot-blockchain applications," *IEEE Transactions on Network and Service Management*, vol. 19, no. 3, pp. 2972–2986, 2022.

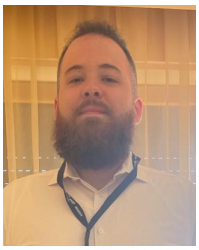
[31] H. Chai, S. Leng, Y. Chen, and K. Zhang, "A hierarchical blockchain-enabled federated learning algorithm for knowledge sharing in internet of vehicles," *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[32] Y. Cheng, D. Wang, P. Zhou, and T. Zhang, "A survey of model compression and acceleration for deep neural networks," *arXiv preprint arXiv:1710.09282*, 2017.

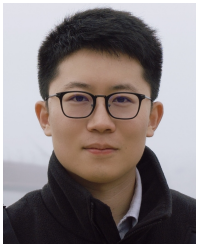
Authorized licensed use limited to: Xi'an Jiaotong-Liverpool University. Downloaded on October 05,2023 at 08:02:57 UTC from IEEE Xplore. Restrictions apply.
© 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Services Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TSC.2023.3318990

Hao Guo received the B.S. and M.S. degrees from the Northwest University, Xi'an, China in 2012, and the Illinois Institute of Technology, Chicago, United States in 2014, and his Ph.D. degree from the University of Delaware, Newark, United States in 2020, all in computer science. He is currently an Assistant Professor with the School of Software at the Northwestern Polytechnical University. His research interests include blockchain and distributed ledger technology, data privacy and security, cybersecurity, cryptography technology, and Internet of Things (IoT). He is a member of both ACM and IEEE.

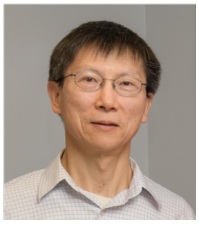


Collin Meese received his B.S. degree in computer science from the University of Delaware in 2020. He is currently working toward the Ph.D. degree at the University of Delaware. His research interests include blockchain, vehicular networks, distributed and high-performance computing, connected and autonomous vehicles, and intelligent civil systems. He is a student member of IEEE.



Wanxin Li (Member, IEEE) received the B.S. degree from the Chongqing University in 2015, and the M.S. and Ph.D. degrees from the University of Delaware in 2017 and 2022, respectively. He is a Lecturer with the Department of Communications and Networking, Xi'an Jiaotong-Liverpool University. His research interests are the privacy and scalability in blockchain, and blockchain-based architecture designs such as connected and autonomous vehicular networks, electronic health records and federated learning.

Mark Nejad is an Assistant Professor at the University of Delaware. His research interests include network optimization, distributed systems, blockchain, game theory, and automated vehicles. He has published more than forty peer-reviewed papers and received several publication awards including the 2016 best doctoral dissertation award of the Institute of Industrial and Systems Engineers (IISE) and the 2019 CAVS best paper award IEEE VTS. His research is funded by the National Science Foundation and the Department of Transportation. He is a member of the IEEE and INFORMS.



Chien-Chung Shen received his B.S. and M.S. degrees from National Chiao Tung University, Taiwan, and his Ph.D. degree from UCLA, all in computer science. He was a research scientist at Bellcore Applied Research working on the control and management of broadband networks. He is now a Professor in the Department of Computer and Information Sciences at the University of Delaware. His research interests include blockchain, federated learning, Wi-Fi, SDN, digital twins, and cybersecurity education. He is a recipient of the NSF CAREER Award and a member of ACM and IEEE.

Authorized licensed use limited to: Xi'an Jiaotong-Liverpool University. Downloaded on October 05,2023 at 08:02:57 UTC from IEEE Xplore. Restrictions apply.
© 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.