This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2020.3040968, IEEE Transactions on Industrial Informatics

A Blockchain-based Deep Learning Approach for Cyber Security in Next Generation Industrial Cyber-Physical Systems

Shailendra Rathore, and Jong Hyuk Park*

Email: {rathoreshailendra2, jhpark1}@seoultech.ac.kr}[1]

***Abstract*—** With the recent development of Internet of Things (IoT) in the next generation Cyber-Physical System (CPS) such as autonomous driving, there is a significant requirement of big data analysis with high accuracy and low latency. For efficient big data analysis, Deep Learning (DL) supports strong analytic capability; it has been applied at the cloud and edge layers by extensive research to provide accurate data analysis at low latency. However, existing researches failed to address certain challenges, such as centralized control, adversarial attacks, security, and privacy. To this end, we propose DeepBlockIoTNet, a secure DL approach with blockchain for the IoT network wherein the DL operation is carried out among the edge nodes at the edge layer in a decentralized, secure manner. The blockchain provides a secure DL operation and removes the control from a centralized authority. The experimental evaluation demonstrates that the proposed approach supports higher accuracy.

Index Terms — IoT, Cyber-Physical Systems, Deep Learning, Blockchain, Security and Privacy

I. Introduction

In recent years, Internet of Things (IoT) has been supported in many Cyber-Physical System (CPS) applications such as augmented reality, virtual reality, and efficient analysis wherein a large amount of data is generated and transferred to the cloud to support analysis and computation of the data, also known as cloud intelligence. These applications require low latency and real-time response [1], [2]. For instance, an autonomous driving application is sensitive to low latency and higher accuracy of data analysis to facilitate accurate and quick decision making in case of a road accident. To satisfy the requirement of higher accuracy and low latency, the application needs higher communication bandwidth and efficient cloud intelligence.

In particular, to support efficient cloud intelligence, Deep Learning (DL) has become the most promising approach in recent years for next-generation CPS [3], [4]. The DL approach works as a strong analytic tool for employing reliable mining (feature extraction and representation) of big unstructured data from IoT, such as speech and images generated from various IoT devices. The mined data is used for various tasks, such as object detection, speech recognition, image classification, and so on. The accuracy and latency of big data analysis using DL rely on the continuous provisioning of application-aware data generated by terminal devices. Fortunately, with the rapid growth of User Equipment (UE), the data collected by the UE exhibit exponential growth; this, in turn, is able to provide the foundation for the requirements in order to obtain cloud intelligence via real-time analysis [5]. Note, however, that cloud intelligence requires the offloading of all data from the UE to the cloud server, further necessitating higher communication bandwidth and high energy consumption and leading to failure to obtain low latency. Moreover, cloud intelligence relies on a centralized architecture wherein the cloud server has full control of the DL model; thus, resulting in a single point of failure [6], [7].

To overcome the problem in cloud intelligence, the DL task can be transferred to the edge computing layer to perform distributed DL, also known as edge intelligence [8], [9]. Edge intelligence supports the distribution of data or computation load from cloud to edge to reduce communication latency. The edge server on the network edge (e.g., small cell base station and router) is deployed with computing capacity and storage to perform the necessary computation. Since the edge server is nearer to the UEs, and only one hop is required in most cases, this leads to the reduction of communication bandwidth and latency [9].

In edge intelligence, however, the cloud server controls the distributed DL task performed by each individual edge node. Therefore, the issue of a single point of failure remains just like in cloud intelligence. On the other hand, edge intelligence might have the effect of an adversarial attack, wherein any entity (cloud or edge server) of the IoT network might act as a malicious or curious entity in the distributed DL task, e.g., an interested server can falsify the overall distributed DL process by implanting the misleading or poisonous training data intentionally. Moreover, an adversary may inject false data at the cloud server, leading to the disruption of the DL task at the edge server in the IoT network [10]. Apart from adversarial attacks, some edge nodes attempt to avoid collaboration in the distributed DL process due to the privacy issue, e.g., in healthcare applications, hospitals or research institutes are unwilling to share the medical data of their patients due to privacy regulations such as HIPAA [11]. On the other hand, in business applications, some companies may be reluctant to participate in collaborative training because they are very

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (NRF-2019R1A2B5B01070416).

Paper no. TII-20-2759. (Corresponding author: Jong Hyuk Park.)

S. Rathore and J. H. Park are with the Department of Computer Science and Engineering, Seoul National University of Science and Technology, Seoul 01811, South Korea (e-mail:,rathoreshailendra@seoultech.ac.kr; jhpark1@seoultech.ac.kr).

concerned about the possible disclosure of their valuable data during distributed training [12]. Consequently, a deficiency of training data leads to a poorly distributed DL model with low accuracy.

Even though an extensive study is being carried out on DL in the IoT network, we identified three serious problems that should be considered to perform an effective DL task on the modern IoT network; 1) Centralized control; 2) Security and privacy (e.g., adversarial attack, training data poisoning); and 3) Low accuracy. Given that, the distributed DL is a widely adopted approach to mitigate the issue of accuracy and latency [13]. In the distributed DL, a server referred to as a parameter server aggregates the local model updates from multiple local parties to prepare a global model update. Each individual party prepares its local model update by performing the DL task on its own training data and uploads the local model update to the parameter server. Upon receiving the local model updates from all parties, the parameter server performs the aggregation of local model updates to obtain a global model update, which is downloaded by each party to update its local model. This process is continued until the training error of the final DL model is lower than a pre-specified threshold. Since learning from multiple parties is combined, this overcomes the deficiency of training data and improves the accuracy. Moreover, each party receives a final global model update on its place, which meets the low latency requirement. Nonetheless, distributed DL cannot protect security and privacy.

On the other hand, a blockchain technology that facilitates secure peer-to-peer connection among the untrusted parties has become a desirable technique to acquire a secure IoT network for next-generation CPS [14], [15]. Recently, Biswas et al. [16] proposed a scalable blockchain scheme that provides a platform of immutable distributed ledger for storing and transferring transactions in a secure, decentralized manner among a group of nodes. Yang et al. [17] studied the integration of blockchain and edge computing to meet performance, scalability, and security requirements in IoT. They examined the impact of blockchain solutions at the network edges (i.e., edge devices) in terms of the computation, storage, and network control to realize the network security, computation, and data integrity verification. Moreover, Nyamtiga et al. [18] proposed a novel approach by converging blockchain with edge computing to address adaptability, integrity, and anonymity issues in IoT. Our recent research [19] introduced a decentralized security architecture for IoT to mitigate the issues of centralization, high latency, the high cost of computation, and storage constraints. Thus, blockchain can also play a significant role in the distributed DL to provide trusted and secure transactions (local and global model updates) among the different parties to mitigate the challenges of security and privacy and remove the control from a centralized authority.

The combination of distributed DL and blockchain can mitigate the existing challenges of edge and cloud intelligence and provide a secure, efficient learning task. Therefore, our research aims to design and develop a decentralized, secure DL approach for the IoT network with the combination of distributed DL and blockchain. The major research contribution of our research is as follows:

- We propose DeepBlockIoTNet, a secure DL approach with blockchain for the IoT network wherein a decentralized DL operation among the edge nodes at the edge layer of the IoT network is supported to mitigate the existing challenges in cloud and edge intelligence.
- An operational design is introduced to deploy distributed DL in the blockchain environment. It facilitates the secure collection and aggregation of the local DL model from multiple edge servers through blockchain transactions to deliver a decentralized, secure DL task.
- In order to validate the practicability and feasibility of the proposed DeepBlockIoTNet, we carried out an experimental analysis by implementing a prototype of it.

The rest of this paper is organized as follows: Section II describes the background and related work of big data analysis in IoT; Section III discusses the design overview of the proposed DeepBlockIoTNet and its operational components; Section IV presents the working mechanism of the proposed DeepBlockIoTNet; Finally, Section V presents the conclusion.

II. Distributed Deep learning

In a big data analysis task, DL provides training generalization and stability and obtains significant scalability. It supports the extraction of nonlinear and complex hierarchical features from highly dimensional training data using multi-layer deep networks consisting of an input layer, output layer, and multiple hidden layers. Each $(l-1)^{th}$ layer computes the output in terms of weight W_l and bias b_l using nonlinear activation function f and delivers it to the next l^{th} layer of the neural network, i.e., $a_l = f(W_l a_{k-1})$ [20]. For a given set of unlabeled training data $X = \{x_1, x_2, \dots x_s\}$, the DL task involves computing a loss function on the given data and minimizing the loss function further to obtain the minimum value of the DL model error on the set of training data, i.e., the gap between the model output value and the target value [21]. The loss function is described using equation 1, where the first term refers to an error of reconstruction defined as the mean of sum-of-square error for s samples of training data, and the second term denotes the employed to alleviate the problem of overfitting in the training process.

$$J(W,b) = \frac{1}{2s}\sum_{k=0}^{s}(||x_k - \widehat{x_k}||)^2 + \frac{\lambda}{2}\sum_{l=1}^{L-1}\sum_{i=1}^{N}\sum_{j=1}^{N+1}\left(W_{ji}^{(l)}\right)^2 \quad (1)$$

Where L and N denote the total number of layers and the number of neurons in each layer.

The minimization of loss function $F(W,B|j)$ is carried out using the stochastic gradient descent (SGD) mechanism [22], wherein the backpropagation method supports the computation of standard gradient $\nabla F(W,b|J)$ using learning rate α. The final value of parameters W,B is computed via averaging. An iteration of updating the operation of weight W and bias b using standard gradient descent over sample i is illustrated in equation 2.

$$W_{ji} \coloneqq W_{ji} - \alpha\frac{\vartheta F(W,b|j)}{\vartheta W_{ji}}$$

$$b_{ji} := W_{ji} - \alpha \frac{\vartheta F(W,b|j)}{\vartheta b_{ji}} \tag{2}$$

High computation overhead is required to perform the abovementioned training process for obtaining a multi-layer, complex DL model. To reduce the overhead, many researchers proposed a distributed DL approach in the past few years [23], [24] wherein a centralized parameter server splits a DL model task among multiple edge nodes by partitioning a whole training dataset into several subsets. Each edge node trains a DL model on its own subset and shares the local update of the model by uploading the parameters (weight W and bias b) to the centralized parameter server. Later, the parameter server prepares a global model update by aggregating the local updates from all edge nodes. Consequently, all edge nodes update their DL model by downloading the global model update from the parameter server. This process is repeated until all edge nodes achieve the final DL models under a certain condition. The overall working mechanism of the distributed DL is illustrated in Algorithm 1.

Algorithm 1: Distributed DL operation

```
1:   Input: $E = \{e_1, e_2, \dots, e_k, \dots, e_n\}$: a set of edge nodes associated with
     a centralized parameter server $P$, $s_k$ : a set of data samples having by
     an edge node $e_k$.
2:   Output: Global model update on $E = \{e_1, e_2, \dots, e_k, \dots, e_n\}$
3:   Process:
4:   Initialize: Training parameters, weight $W_{jk}$, bias $b_{jk}$ broadcasted
     from the parameter server $P$ to all edge nodes $E$
5:   For each edge node $e_k$ in the $E$ do in parallel:
6:       //Run SGD over data samples $s_k$
8:       Compute $W_{jk} := W_{jk} - \alpha \frac{\vartheta F(W,b|j)}{\vartheta W_{jk}}$
9:       Compute $b_{jk} := W_{jk} - \alpha \frac{\vartheta F(W,b|j)}{\vartheta b_{jk}}$
10:      Compute local model update $\Delta W_{jk}$ and $\Delta b_{jk}$
11:      Upload $\Delta W_{jk}$ and $\Delta b_{jk}$ to parameter server $P$
12:  End
13:  Update $W_{jk}$, $b_{jk}$= $W_{jk} + \Delta W_{jk}$, $b_{jk} + \Delta b_{jk}$ // Apply the update
     parameters on the parameter server
14:  Repeat (4)
15:  End
```

Note, however, that the distributed DL has a centralized control issue wherein the parameter server has full control over the distributed DL process, which can create the problem of single point of failure. In addition, security and privacy are also major issues, with any entity (parameter server or edge node) possibly falsifying the overall distributed DL process for malicious and curious intent. Recently, Weng et al. [25] introduced DeepChain; a privacy-preserving distributed DL with blockchain technology wherein incentive mechanism and cryptographic primitives (i.e., a homomorphic encryption technique, zero-knowledge proof) are used to achieve computation auditability and data confidentiality in distributed DL. However, the incentive mechanism is challenging in the deployment of DeepChain on IoT platform. To mitigate these challenges, we employ the blockchain technology in the distributed DL and introduce a decentralized, secure DL approach for IoT, which is different from DeepChain system in the following aspects. Unlike the DeepChain, the collected data at each node in the proposed approach is measured by the corresponding edge node; the gathered information is independent and unrelated to earlier measurements. Also, data verification in the DeepChain requires to ensure that all parties and workers have adequate balance (i.e., DeepCoin) to manage the blockchain transaction. In contrast, data verification in our proposed approach is carried out using the voting process, i.e., task corresponding to DeepCoin deposit checking are not needed in the proposed approach. In other words, the proposed approach does not have a Merkle tree situation [14], which provides a faster and lightweight chain connection in the proposed approach, unlike the DeepChain. The proposed approach supports algorithmic improvement to develop a reliable and fast mining method wherein competition among all miners in solving a puzzle and no incentive as a reward is given to the miner who solves the puzzle first.

In the subsequent section, we will discuss the design overview and operational components of the DeepBlockIoTNet.

III. Proposed DeepBlockIoTNet Architecture

The DL approach proposed in this paper supports a decentralized, secure big data analysis task wherein a distributed DL operation at the edge layer of the IoT network is carried out in a secure, decentralized manner using blockchain technology. Typically, three basic processes are compiled for big data analysis in the modern IoT network: data gathering and pre-processing at the remote server from IoT devices, analysis of processed data using intelligent learning algorithms; and remote control of IoT devices using the analyzed data. The current big data analysis approaches provide centralized control and management but have a tremendous amount of data being exploited maliciously by an adversary. The proposed DeepBlockIoTNet remarkably lowers the possibility of data being manipulated in an adversarial manner by facilitating a secure, decentralized DL approach. In this section, we discuss the design overview of the proposed DeepBlockIoTNet and its operational components.

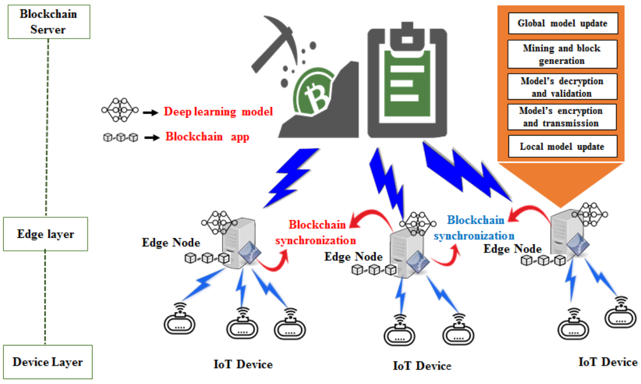


Fig. 1: Design overview of the proposed DeepBlockIoTNet

A. *DeepBlockIoTNet Design Overview*

Fig. 1 illustrates the design overview of the proposed DeepBlockIoTNet consisting of three layers: device, edge, and cloud layers. The device layer supports several widely distributed sensor nodes associated with various smart devices responsible for monitoring various activities and environments in the public infrastructure and generates a massive amount of big data. As described in research [26], the edge layer consists of several edge nodes, each of which is subject to processing and analysis of the generated data by its associated sensors (at its premises). To analyze the big data, each edge node is configured with DL and blockchain applications. The blockchain server is configured to support blockchain services. The peer-to-peer interactions among the edge nodes are supported via transactions implemented by employing blockchain technology.

B. *Operational Components of the DeepBlockIoTNet*

In this subsection, we describe the operational modules to implement the proposed DeepBlockIoTNet. Fig. 2 illustrates the operational flow of DeepBlockIoTNet, wherein edge nodes interact with each other via the blockchain server to accomplish a secure, decentralized DL operation. The operational design mainly includes six components along the path between the edge nodes and the blockchain server, as shown in Fig. 2, and it is explained as follows:

1) Learning agent: In the proposed system, a learning agent is an edge node participating in the decentralized DL because he/she is not able to prepare an accurate DL model alone due to resource constraints such as lack of training data or scarcity or shortage of computational power. This agent is responsible for preparing a local model employing a DL approach on his/her private data and sharing the local model in the decentralized DL task.

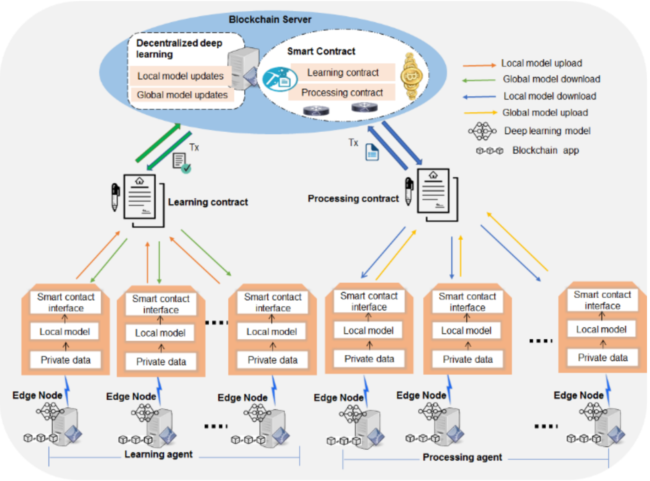


Fig. 2: Operational flow of the proposed DeepBlockIoTNet

2) Processing agent: An edge node is responsible for the processing and aggregation of the local models from the learning agents to prepare a global model (decentralized DL) with higher accuracy. The multiple processing agents who have enough computation power compete in the blockchain network to carry out the decentralized DL task. The agent who finish the decentralized DL task first uploads the global model to the blockchain server, which is then downloaded by the learning agents.

3) Smart Contract: The proposed system states all the required policies and rules for its operation in a smart contract, which includes two key modules: learning contract and processing contract. The learning contract helps the learning agents upload and download operations, i.e., the agents upload the parameters of their local models referred to as local model updates to the blockchain server and download the processed and aggregated local model updates referred to as global model update from the blockchain server. The learning contract defines various policies and rules., i.e., uploading function for local model update and downloading function for the global update. On the other hand, the processing contract supports the processing agents in uploading and downloading operations, i.e., the agents download the list of local model updates from the blockchain server for further processing and aggregation and upload the global model update to the blockchain server. The processing contract defines various policies and rules., i.e., downloading function for the list of local model updates and uploading function for a global model update.

4) Smart contract interface: It provides an interface for connecting the edge nodes and the blockchain server to the smart contract. It is responsible for triggering the operations and the activities of the smart contracts at the edge nodes, such as registration of an edge node in the blockchain network, sharing of local model updates, communication and synchronization among edge nodes and blockchain server, and edge node requesting for decentralized DL (global model update). A Javascript-based API known as Web3 protocol is employed in the proposed system to configure the smart contract interfaces for the edge nodes. A smart contract interface can call the functions and perform the rules encoded in the contract on behalf of the IoT device.

5) Blockchain network: We employ a private blockchain in our proposed DeepBlockIoTNet to support a decentralized, secure DL task in IoT. The private blockchain does not support pure peer-to-peer control, with the application owners of the blockchain such as stakeholders having control over the blockchain network; hence provides higher throughput at low latency, which is required for the big data analysis task in IoT. Moreover, the private blockchain provides a less resource-intensive mining task and supports the mining task to an edge node with resource constraints in the DeepBlockIoTNet. In blockchain deployment, the learning agent acts as a blockchain client, and a processing agent carries out the mining task. In particular, a resource-intensive task of block generation and mining is executed by the processing agent. The learning agents support the local DL learning task and offload the global model generation task to the processing agents.

In the DeepBlockIoTNet, an Ethereum private blockchain platform act as a blockchain server that typically supports the blockchain services, execution of smart contracts, and recording the activities of edge nodes. The blockchain server interacts with the edge nodes and supports blockchain services using two key operations. Primarily, the collection of all the transactions among edge nodes and smart contracts is configured at the blockchain server, which supports the generation of a new block in the blockchain network. Second is the recording of all the activities in the blockchain network, such as information about the mining blocks and requesting and logging edge nodes.

IV. Working Mechanism of DeepBlockIoTNet

In the proposed DeepBlockIoTNet, all local and global model updates from the learning and processing agents, respectively, are eventually uploaded to a distributed ledger (i.e., Blockchain server) of the blockchain network in the form of a connected block, which presents a distributed form in each agent's memory. Before uploading, several processes are needed to ensure the model's accuracy; model's broadcasting; model's validation using a voting scheme; model's aggregation in block and block mining task; validation of the mining outcomes using voting scheme; and finally, synchronization of the distributed ledger. This section describes the operational details of DeepBlockIoTNet, which typically involves the model's transmission (encryption/decryption), validation, and uploading or recording.

A. Model's Encryption and Transmission

To support the model's encryption and transmission, a public key and a private key are assigned to each edge node in the IoT network. A public key signifies publicly available and accessible information in the network with regard to an edge node. Conversely, a private key implies private information about an edge node that supports the validation of the node's operation and identity. Since all edge nodes are distributed in a blockchain network, the associated data (DL model update) with an edge node is required to encrypt and then transmit to all connected nodes (broadcasting). The process of the model's encryption and broadcasting is depicted in Fig. 3.

The data associated with each edge node consists of general information and transmitted data. The general information includes the private key of the edge node, a public key of all connected nodes, candidate blocks, and preset consensus. The transmitted data consists of a local model update and its digital signature. To generate the digital signature, each edge node follows the two steps operation: 1) The edge node takes the newly collected local update and put it through a Secure Hash Algorithm (SHA) to produce a message digest (i.e., a unique string of characters); 2) The produced message digest is then digitally signed using the public key encryption algorithm and the node's private key. Finally, the local model update and its digital signature are sent to all connected nodes in the form transmitted data through the communication network. In our proposed approach, we used Elliptic Curve Digital Signature Algorithm (ECDSA) for public-key cryptography due to its advantage of faster computations, less storage space, and shorter key size over the other traditional algorithms such as RSA. The broadcasted data by edge nodes are handled using the model's decryption and validation component, which is described in the subsequent subsection.

B. Model's Decryption and Validation

The broadcast data by an edge node is received by all other connected edge nodes in an encrypted form that should be decrypted and verified. As shown in Fig. 3, a recipient hashes the local model update into message digest-1 and decrypts the digital signature into message digest-2 via the sender's public key. The received data is verified as correct if message digest-1 and message digest-2 are equal (True). In contrast, the received data is regarded as incorrect (false) due to issues of data consistency and integrity presence in the broadcasting process. For instance, the transmitted data might be inconsistent, delayed, tampered with, or even discarded between message digest-1 and message digest-2.

A distributed voting scheme (majority voting mechanism) is used in the proposed DeepBlockIoTNet, wherein a chance is given to each recipient edge node to vote (true or false) for verifying the consistency and integrity of the received data. As illustrated in Figure 3, the received data is verified as correct or accepted only when a positive majority vote (True) is obtained among the edge nodes.

Suppose there are n edge nodes that deliver their votes to verify the received data from an edge node, given that v is the total number of positive votes (true) (the negative vote is $\sim v$) where $v \leq n$. Then, the data is considered accepted when the following condition is satisfied:

$$\frac{v}{n} > \theta \qquad (3)$$

Where θ stands for the threshold value that must be $0.5 < \theta < 1$ to ensure the majority voting for an accepted data. As a result of validation, all accepted data over a certain time period are recorded in the form of a candidate block in the distributed ledger at the blockchain server (Z).

All the verified local model updates over a certain time period are used to generate blocks in the blockchain using block mining and generation component, which is described as follows:

C. Mining and Generation of Blocks

All the verified local model updates over a certain time period (called an epoch) are packages in a candidate block to exchange the local model updates truthfully among all the edge nodes in the IoT network. The generated blocks in the consecutive epochs are cryptographically connected block by block in a distributed blockchain network. To link the blocks cryptographically, various secure hash algorithms (SHAs) applying a one-way hash function on the message in the candidate block to generate a message digest can be employed, such as SHA-512, SHA-384, SHA-256 [27].

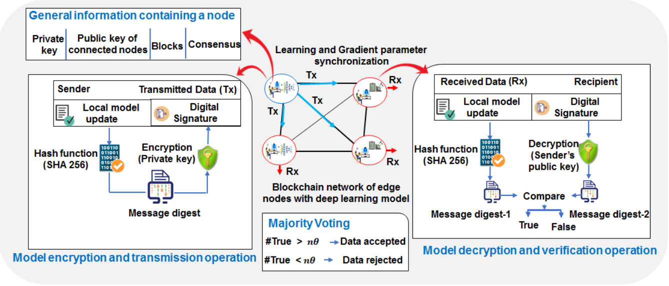

General information containing a node
Private key
Public key of connected nodes
Blocks
Consensus
Learning and Gradient parameter synchronization
Sender
Transmitted Data (Tx)
Local model update
Digital Signature
Hash function (SHA 256)
Encryption (Private key)
Message digest
Model encryption and transmission operation
Tx
Tx
Tx
Rx
Rx
Rx
Blockchain network of edge nodes with deep learning model
Majority Voting
#True > $n\theta$ → Data accepted
#True < $n\theta$ → Data rejected
Received Data (Rx)
Recipient
Local model update
Digital Signature
Hash function (SHA 256)
Decryption (Sender's public key)
Compare
Message digest-1
Message digest-2
True
False
Model decryption and verification operation

Fig. 3: Model's encryption, transmission, decryption, and validation operation

The SHA-256 function is used in the proposed DeepBlockIoTNet for the blockchain ledger generation and mining process. Employed in many bitcoin systems, it supports intermediate computational complexity. Note, however, that the proposed DeepBlockIoTNet can be easily configured with other hash functions.

A block in a blockchain ledger consists of two parts: body and header. In the traditional blockchain architecture [14], the body consists of a set of verified transactions. On the other hand, in DeepBlockIoTNet, a set of verified local model updates of the edge nodes in E over an epoch (i.e., $\{\Delta W_{jk}, \Delta b_{jk}\}_{e_k \in E}$ for device e_k in E over an epoch) are stored in the body. Consequent with the structure [27], the header consists of block number, previous hash result, nonce solution (output value of the Proof of Work (PoW)), timestamp, and current hash result.

In the block generation process, the SHA-256 algorithm employs a one-way hash function to find the output 32-bit words with elements from {0, 1,…, 9, A, B,…, F} and consists of two phases: pre-processing and hash computation.

The pre-processing phase is responsible for processing all the information to generate a candidate block and summarize it in message M as follows:

$$M = \mathcal{B} + \mathcal{L} + \mathcal{H}_p + Nonce + T \quad (4)$$

Where $\mathcal{B}$, $\mathcal{L}$, and $\mathcal{H}_p$ stand for block number, list of verified local model updates (also known as a global model update), and previous hash result, respectively. $Nonce$ and T represent the random number and timestamp, respectively.

Assume that a list of local updates from all edge nodes over an epoch has been verified and packaged into the body of the b^{th} block as data content. Then, a puzzle problem is solved by some edge nodes to determine an appropriate value of the nonce for the given data content of the b^{th} block, current timestamp, and hash results of the $(b-1)^{th}$ block. This puzzle-solving task is known as mining, and the edge nodes contributing in the mining task are called processing agents (known as miners in traditional blockchain architecture [28]). Furthermore, the hash computation phase is responsible for the generation of the puzzle problem wherein the message digest $Puzzle_{hash}$ is produced by applying SHA-256 twice to input message M to support extra security measure as follows:

$$Puzzle_{hash} = hash(\text{SHA256}, hash(S\text{HA256}, \text{M})) \quad (5)$$

Algorithm 1: DeepBlockIoTNet operation

```
1:  Input: $E = \{e_1, e_2, \dots, e_k, \dots, e_n\}$: a set of edge nodes associated with a Blockchain server $Z$, $s_{e_k}$ : a set of data samples having by an edge node $e_k$, $(pk_{e_k}, pu_{e_k})$: a pair of private and public key for an edge node $e_k$, $\theta$: Constant for majority voting mechanism, $\mathcal{B}$: Block number, $\mathcal{H}_p$: Previous hash result, $T$: Timestamp
2:  Output: Global model update on $E = \{e_1, e_2, \dots, e_k, \dots, e_n\}$
3:  Process:
4:  Initialize: Training parameters, weight $W_{jk}$, bias $b_{jk}$ broadcasted from the blockchain server $Z$ to all edge nodes $E$ , Initialize: True =0
/* Local model computation */
5:  For each edge node $e_k$ in the $E$ do in parallel:
6:      //Run SGD over data sample $s_k$
8:      Compute $W_{jk} := W_{jk} - \alpha \frac{\vartheta L(W,b|j)}{\vartheta W_{jk}}$
9:      Compute $b_{jk} := W_{jk} - \alpha \frac{\vartheta L(W,b|j)}{\vartheta b_{jk}}$
10:     Compute local update $\Delta W_{jk}$ and $\Delta b_{jk}$
11:     Generate: $\boldsymbol{Sign}_{e_k}$ = Encrypt($pk_{e_k}$, (Hash ($\Delta W_{jk}$, $\Delta b_{jk}$))
12:     Broadcast (($\Delta W_{jk}$, $\Delta b_{jk}$), $Sign_{e_k}$) to blockchain server $Z$ /* Transmitted data*/
13:     Receive (($\Delta W_{jk}$, $\Delta b_{jk}$), $Sign_{e_k}$) by $(E - e_k)$ edge nodes
                                                        /* Received data*/
        /* Local update verification */
14:       For each edge node $(E - e_k)$ in the $E$ do in parallel:
15:           Compute $Mdigest_1 = Hash(\Delta W_{jk}, \Delta b_{jk})$
16:           Compute $Mdigest_2$= Decrypt( $pu_{e_k}$,$Sign_{e_k}$)
17:           If $(Mdigest_1 = Mdigest_2)$ Then
18:               Update True = True+1
19:       End
20:       If $(True > n\theta)$ Then
21:           Update $\mathcal{L} \longleftarrow (\Delta W_{jk}, \Delta b_{jk})$
              /* $\mathcal{L}$: List of verified local model updates*/
22:       Else
23:           Discard $(\Delta W_{jk},\ \Delta b_{jk})$
24: End
/* Block Mining and Generation */
25: $Puzzle_{hash} = Hash(SHA256, Hash(SHA256, \mathcal{L}))$
                                          /* Nonce computation */
26: Generate $M = \mathcal{B} + \mathcal{L} + \mathcal{H}_p + Nonce + T$
27: Broadcast $M$ to all edge nodes in $E$
28: Receive $M$ by $E_p$ edge nodes /* $E_p = E - E_l$ : a set of processing agents for a given set of learning agents $E_l$ */
29: For each edge node in $E_p$ do in parallel:
30:     Solve $Puzzle_{hash}$
31: End
32: Select edge node $e_i$ who first solve $Puzzle_{hash}$
33: Verify $Nonce$ value using majority voting mechanism
34: Add Block $M$ to Blockchain server $Z$
/* Global model computation */
35: For each edge node $e_k$ in the $E$ do in parallel:
36:     Download block M form the blockchain server $Z$
37:     Compute Global model update using the aggregated local model updates in the block $M$ (i.e., equation 7)
38: End
39: Repeat (5)
```

The puzzle problem is defined as finding an appropriate value of the $Nonce$ to generate the lesser value of $Puzzle_{hash}$ then a given target Q, illustrated as follows:

$$Puzzle_{hash} \leq Q \quad (6)$$

The puzzle problem above is solved using a brute force way, which is highly computationally intensive. The computing difficulty of $Puzzle_{hash}$ relies on the value of Q (i.e., the number of leading zeros in Q). Generally, the longer the length of leading zeros in Q raises the difficulty level of generating $Puzzle_{hash}$ and the more processing time and computational resources are required to solve the puzzle. In order to alleviate this problem in our proposed approach, we adjust the difficulty to a significantly low level by setting the smaller length of leading zeros in Q to support quick mining.

Some or even all edge nodes (processing agents) with enough computation power can participate in solving the puzzle problem. A processing agent who determines the value of nonce first disseminates it to other connected processing agents in order to verify the correctness of the $Nonce$ value. The connected processing agents give their feedback on verification in terms of voting (True or False), which is further evaluated by again employing a distributed voting mechanism. As illustrated in equation 3, once enough processing agents give positive responses to the $Nonce$ value, the candidate block is permitted to link in the distributed ledger at the blockchain server Z cryptographically. All the processing agents are compelled to

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2020.3040968, IEEE Transactions on Industrial Informatics

halt their nonce finding tasks, and a generated candidate block is added to their local ledger. Subsequently, the distributed ledger at the blockchain server is updated with a newly generated block synchronously, and all processing agents move forward to the generation of the next block.

On the other hand, some or even all edge nodes (learning agents) who are willing to prepare a global DL model can download the newly generated candidate block from the distributed ledger at the blockchain server Z. They locally compute a global model update by using the aggregated list of verified local model updates in the newly generated block described as follows [13]:

$$W^{(p)} = W^{(p-1)} - \frac{1}{n}\sum_{k=1}^{n}(\Delta W_{jk},\ \Delta b_{jk}) \qquad (7)$$

Where $W^{(p)}$ and $W^{(p-1)}$ represent the global model updates at the p^{th} and $(p-1)^{th}$ epoch, respectively.

However, two edge nodes may find the correct nonce at the same time or another edge node successfully generate another block within the propagation delay of the firstly generated block; then this secondly generated block might be incorrectly added by some edge nodes to their local ledgers, refer to forking. To prevent from forking, an Acknowledgement (ACK) signal is used wherein an ACK signal is broadcasted when each edge node finds no forking occurrence. Every edge node waits until getting ACK signals from all other edge nodes otherwise the operation of block generation restarts again.

V. Experimental Analysis

A. DeepBlockIoTNet testbed setup:

In order to evaluate the Quality of Service (QoS) of the proposed DeepBlockIoTNet in practical applications, we implemented a prototype model using Raspberry Pi's with Raspbian Operating System (OS) as IoT devices and workstations with Intel processor as edge computing nodes. DeepBlockIoTNet was set up with 30 workstations with several Raspberry Pi's. Regarding the blockchain configuration, Go-*ethereum* was employed as a blockchain, and smart contracts were written with solidity language and deployed using a Truffle development suite. Our blockchain is configured on the Ethereum platform [28] which is initialized by default to sync with a live public network. However, our proposed DeepBlockIoTNet is presently designed for the experimental purpose, so we configure ethereum for use on a private network on the university campus. Moreover, Node.js was employed to support an interface for interaction between blockchain and IoT applications. The PoW was implemented using C++ on Intel Core i7 processor to examine the processing time of solving the puzzle with several difficulties. We found that solving PoW with 6 leading zeros in target hash Q takes 9.09ms average lowest time to synchronize a new block by IoT edge device. Therefore, in the experiment, we adjusted the difficulty to 6 leading zeros in Q. On the other hand, a DL was built with Tensorflow version 1.7.0, NumPy version 1.14.0, and python version 3.6.4. Each edge computing node was configured for Go-ethereum and DL operations.

In the experimental setup, some edge nodes were operated as learning agents responsible for generating and sending the local model updates to the blockchain server in the form of blockchain transactions and getting the list of verified local model updates from the blockchain server to obtain global updates. Conversely, some edge nodes were configured as a processing agent responsible for validation and block mining task. The object detection problem is widely discussed by many researchers, such as indoor guarding, crowd control, and city surveillance [29-32], requiring a highly accurate result at low latency. For object detection, a well-known MS COCO dataset containing 118,000 training instances and 46,000 validation instances of 80 different object classes [33] was used. The overall dataset was divided into subsets and assigned to edge nodes. In our experimental evaluation, we assume 30 edge nodes (e-1, e-2, e-3, e-4 ,…., e-29, e-30), each of them containing a distinct subset (i.e., trainset=118,000/30=3,933, valset=46,000/30=1533) of the whole dataset. Some of the edge nodes (i.e., 10 edge nodes) acted as a processing agent. Each edge node prepared a trained model.

Note: In the DeepBlockIoTNet, the memory of an edge node occupied by the distributed ledger Z is released periodically. With the continuous operation of the DeepBlockIoTNet (i.e., store the partial neural network model on blockchain and performing the training and inference over time), the distributed ledger at the blockchain server will eventually be large. For instance, if the size of the block header is 0.1 KB, the body is 1 KB, and the generation rate of blocks is one block per minute. Then, the size of the ledger after one year would be (1+0.1) ×60×24×365 = 578160 KB = 579MB (approximately). In this case, recycling the memory space is necessary and can be obtained by freeing up the memory of the edge node occupied by ledger on an annual basis.

B. Evaluation Results:

For object detection, video frames (objects) were captured with a camera module integrated with Raspberry Pi in 60Hz frequency and 1080p resolution and processed further using the DeepBlockIoTNet system, which gives the detection results with boxes over the detected objects. The feasibility of object detection in the multiple Raspberry Pi's settings was evaluated in terms of four metrics: mean precision accuracy, latency delay, privacy, and security analysis that were our considerations for proposing DeepBlockIoTNet to achieve high-assurance CPS [34], [35].

Mean precision accuracy: Fig. 4a shows the accurate measurements of the object detection process via DeepBlockIoTNet. It can be seen in Fig. 4a that the accuracy increases in proportion to the number of edge nodes. Generally, an individual local model update is prepared by each edge node employing the DL operation on its own private data (a subset of 3,933 instances), and this local model update is shared for the distributed DL in DeepBlockIoTNet where a list of verified local updates is produced for the computation of global model update at each edge node. The participation of a greater number of edge nodes obviously leads to an increment in the size of the training dataset (n*3,933, where n is the number of edge nodes). Consequently, the list of local model updates to compute a global model update increase that further provides an escalation in the overall accuracy of the global model update. Moreover, we compared the accuracy of object detection process in the case of "Without DeepBlockIoTNet" wherein each edge node downloads a global model update from a cloud server (a distributed DL) and in the "With DeepBlockIoTNet" case wherein each edge node computes the global model update by downloading a list of verified local model updates from a blockchain server (an integration of blockchain and distributed DL). As illustrated in Fig. 4a, "With DeepBlockIoTNet" (blue color) obtained more accurate results compared to "Without DeepBlockIoTNet" (dark yellow color).

Latency delay: We measured a variation in the object detection time against the total number of edge nodes to demonstrate the latency improvement using DeepBlockIoTNet. The detection time was obtained using $T_{With\, DeepBlockIoTNet} = T_{BC\_global} + T_{edge}$ and $T_{Without\, DeepBlockIoTNet} = T_{global} + T_{cloud}$, where T_{BC_global} stands for the time required to generate a global model update using blockchain operation, which involves the time to sign, hash, propose transactions, and append and retrieve blocks in the blockchain of "With DeepBlockIoTNet". T_{edge} represents an average communication delay in downloading the global model update from the edge node to the IoT device in "With DeepBlockIoTNet." Also T_{global} and T_{cloud} denote the time required to generate global model updates using conventional distributed learning at the centralized cloud server and an average communication delay to download the global model update from cloud to IoT device, respectively in "Without DeepBlockIoTNet." As illustrated in Fig. 4b, there is always an increment in the detection time for both "With DeepBlockIoTNet" and "Without DeepBlockIoTNet" cases. Note, however, that shorter detection time was realized "With DeepBlockIoTNet" than "Without DeepBlockIoTNet." This could be attributed to the fact that, in "With DeepBlockIoTNet," a global model update for object detection is computed at each edge node close to the IoT device, which leads to shorter detection time. On the other hand, in the case of "Without DeepBlockIoTNet," a global model update for object detection is computed at the cloud server and distributed among the edge nodes, causing an extra communication delay to transfer the global update from cloud to edge nodes (i.e., $T_{edge} < T_{cloud}$) and increase the detection time. Here, It should be noted that the extra communication delay is significantly higher than the delay gained by blockchain operation $(i.e., (T_{cloud} - T_{edge}) > (T_{BC\_global} - T_{global}))$.

Computational Overhead: In order to validate the computational feasibility of DeepBlockIoTNet, we evaluated the computational overhead at each edge node in terms of memory and CPU. As illustrated in Fig. 4c, the edge nodes in "With DeepBlockIoTNet" consume additional CPU (lies between 4.1 to 5.5) and memory (lies between 8.1 to 8.8) as compared to that of "Without DeepBlockIoTNet". It can be due to the additional overhead of blockchain (i.e., Consensus process, Block generation and mining) operation at each edge node in DeepBlockIoTNet. Since DeepBlockIoTNet outperforms over the conventional distributed learning in terms of accuracy and latency, slightly additional overhead can be considered acceptable.

Security and privacy: DeepBlockIoTNet consider security and privacy to be an important factor. As we discussed in the introduction section, the existing distributed DL operation may suffer from adversarial attacks [10] that can be divided into the following three categories:

Attack category-1: Compromise the local model update.

Attack category-2: Implant misleading false model's update deliberately during the model's update transmission to/from the central server.

Attack category-3: Compromise the central server database via a data poisoning attack.

We compared the probability of occurrence of these three categories of attacks in both "With DeepBlockIoTNet" and "Without DeepBlockIoTNet" cases. The comparison is depicted in Fig. 5. To compute the probabilities, we consider n edge nodes. All categories of attacks are considered independent of each other to carry out successful attacks.

In "Without DeepBlockIoTNet", the total probability of above mentioned all categories of attacks (i.e., represented as $Pr_{[WoD]}$) can be considered into four fractions and represented as follows:

$$Pr_{[WoD]} = Pr_{[WoD\_1]} + Pr_{[WoD\_2]} + Pr_{[WoD\_3]} + Pr_{[WoD\_4]} \quad (8)$$

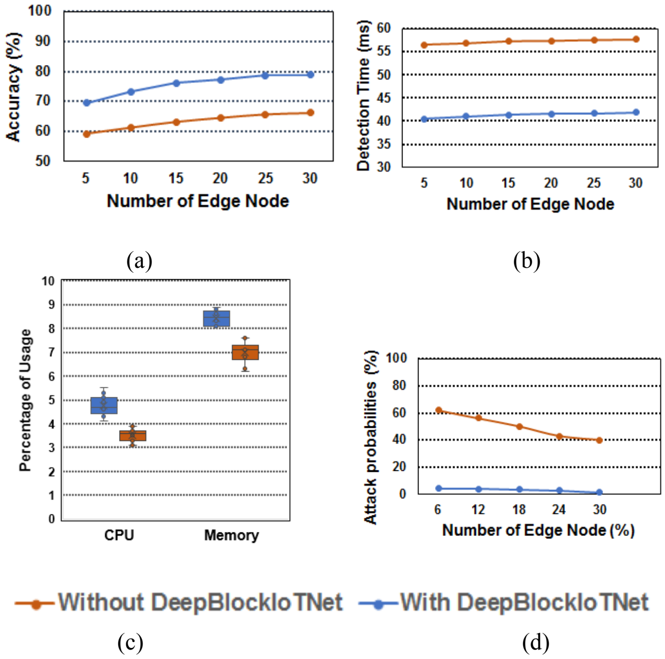
Here, $Pr_{[WoD\_1]} = \frac{1}{4}\prod_{k=1}^{n} \rho_k$ represents the probability of successfully hacking the corresponding n edge nodes in the attack category-1, Since the probability of successful hacking of each edge node k by the attackers represented as ρ_k (i.e., $0 \leq \rho_k \leq 1$) is independent, $Pr_{[WoD\_1]}$ can be calculated by multiplying the individual probability of successful hacking of all edge node n $(\text{i.e.}, \rho_1, \rho_1, \dots \rho_k, \dots, \rho_n)$ and dividing by 4 as total probability is considered into four fractions.

Similarly, we can calculate $Pr_{[WoD\_2]} = \frac{1}{4}\prod_{k=1}^{n} \sigma_k$ that denotes the probability of hacking the corresponding transmission medium by falsifying the local model's updates from edge nodes to the central server (attack category-2). Here, the probability of implanting of the false data into model's update by attackers for each edge node k is independent and represented as σ_k (i.e., $0 \leq \rho_k \leq 1$). Also, $Pr_{[WoD\_3]} = \frac{1}{4}\prod_{k=1}^{n} \sigma_k$ that signifies the probability of hacking the corresponding transmission medium by falsifying the global model's updates from the central server to the edge nodes (attack category-2).

In addition, $Pr_{[WoD\_4]} = \frac{1}{4}\chi$ wherein χ is assumed to be the probability of compromising the central server in the attack category-3. Then, the overall probability of successfully hacking the "Without DeepBlockIoTNet" process can be computed as follows:

$$Pr_{[WoD]} = \frac{1}{4}\prod_{k=1}^{n} \rho_k + \frac{1}{4}\prod_{k=1}^{n} \sigma_k + \frac{1}{4}\prod_{k=1}^{n} \sigma_k + \frac{1}{4}\chi \quad (9)$$

In "With DeepBlockIoTNet," an attacker can carry out all three categories of attacks. Like $Pr_{[WoD]}$, we can consider total probability to successfully hack the "With DeepBlockIoTNet"



(a) (b)

(c) (d)

Fig. 4: DeepBlockIoTNet performance in terms of : a) Accuracy; b) Latency delay; c) Computational overhead; d) Security measurement

process into four fractions (i.e., represented as $Pr_{[WD]}$) and computed as follows:

$$Pr_{[WD]} = Pr_{[WD\_1]} + Pr_{[WD\_2]} + Pr_{[WD\_3]} + Pr_{[WD\_4]} \qquad (10)$$

Here, $Pr_{[WD\_1]} = \frac{1}{4}\prod_{k=1}^{n}\phi_k \times \frac{1}{4}\prod_{k=1}^{n}\bar{\rho}_k$ denotes the probability of attack category-1, wherein the attacker needs to steal the key's information and hack each edge node to carry out successful attacks. The probability of stealing of the key's information for each node k by attackers is independent and represented as ϕ_k (i.e., $0 \le \phi_k \le 1$). The probability of successful hacking of each edge node by attackers is independent and represented as $\bar{\rho}_k$ $(i.e., 0 \le \bar{\rho}_k \le 1)$. Then, $Pr_{[WD\_1]}$ can be mathematically calculated by multiplying the individual probability of stealing the key's information of all edge node $(i.e., \phi_1, \phi_1, \dots \phi_k, \dots, \phi_n)$, successful hacking of all edge node n $(i.e., \bar{\rho}_1, \rho_2, \dots \bar{\rho}_k, \dots, \bar{\rho}_n)$ and dividing by 4 as total probability is assumed into four fractions.

Similarly, we can compute $Pr_{[WD\_2]}$ that represents the probability of hacking the corresponding transmission medium by falsifying the local model's updates from edge nodes to the blockchain server (attack category-2). There are $n(n-1)/2$ transmission medium. Let the probability of implanting the false data in the transmission medium be independent and represented as $(\bar{\sigma}_1, \bar{\sigma}_1, \dots \bar{\sigma}_k, \dots, \bar{\sigma}_{n(n-1)/2})$, where $0 \le \bar{\sigma}_k \le 1$. Since DeepBlockIoTNet employs the threshold ($0.5 < \theta < 1$ as described in equation 3) for the voting mechanism, the attacker needs to hack the $v = \theta \cdot n(n-1)/2$ transmission medium for successful attacks. Furthermore, the attacker needs to steal n pairs of key information to encrypt the false injected data. Then, $Pr_{[WD\_2]} = \frac{1}{4}\prod_{k=1}^{v}\bar{\sigma}_k \times \frac{1}{4}\prod_{k=1}^{n}\phi_k$. Also, $Pr_{[WD\_3]}$ that indicates the probability of hacking the corresponding transmission medium by falsifying global model's updates from the blockchain server to the edge nodes (attack category-2) can be calculated same as $Pr_{[WD\_2]}$ and defined as $Pr_{[WD\_3]} = \frac{1}{4}\prod_{k=1}^{v}\bar{\sigma}_k \times \frac{1}{4}\prod_{k=1}^{n}\phi_k$.

In addition, $Pr_{[WD\_4]}$ implies the probability of attack category-3 wherein to compromise the blockchain server, an attacker should hack at least $v = \theta \cdot n(n-1)/2$ edge nodes and steal n pair of the key's information to falsify the majority voting process. Therefore, the probability of successful attacks in the attack category-3 is $Pr_{[WD\_4]} = \frac{1}{4}\prod_{k=1}^{n}\phi_k \times \frac{1}{4}\prod_{k=1}^{v}\bar{\rho}_k$.

The overall probability of successfully hacking the "With DeepBlockIoTNet" process is denoted as $Pr_{[WD]}$ and can be computed as follows:

$$Pr_{[WD]} = \left(\frac{1}{4}\prod_{k=1}^{n}\phi_k \times \frac{1}{4}\prod_{k=1}^{n}\bar{\rho}_k\right) + \left(\frac{1}{4}\prod_{k=1}^{v}\bar{\sigma}_k \times \frac{1}{4}\prod_{k=1}^{n}\phi_k\right) + \left(\frac{1}{4}\prod_{k=1}^{v}\bar{\sigma}_k \times \frac{1}{4}\prod_{k=1}^{n}\phi_k\right) + \left(\frac{1}{4}\prod_{k=1}^{n}\phi_k \times \frac{1}{4}\prod_{k=1}^{v}\bar{\rho}_k\right) \qquad (11)$$

It can be easily seen in equations 9 and 11 that the overall probability of successful attacks depends on the number of edge nodes (n). We simulate equations 9 and 11 by setting the value of n in the range of (6, 12, 18, 24, 30). We choose the value of other variables based on the difficulty of attacks. We set value $\rho, \sigma, \bar{\rho}, \bar{\sigma}$, and ϕ of in the range of [0.9, 1] and the value of χ in the range of [0, 0.1]; the value of threshold θ is arbitrarily selected from the range [0.5, 1].

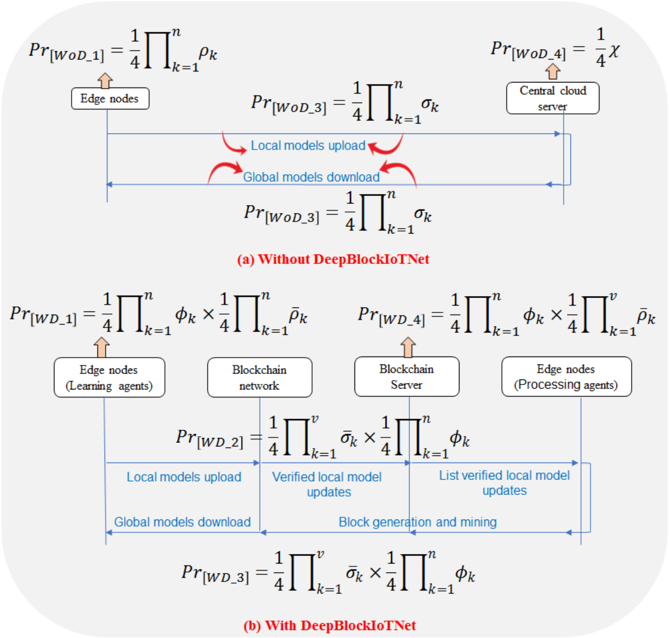

$Pr_{[WoD\_1]} = \frac{1}{4}\prod_{k=1}^{n}\rho_k$
$Pr_{[WoD\_4]} = \frac{1}{4}\chi$
Edge nodes
$Pr_{[WoD\_3]} = \frac{1}{4}\prod_{k=1}^{n}\sigma_k$
Central cloud server
Local models upload
Global models download
$Pr_{[WoD\_3]} = \frac{1}{4}\prod_{k=1}^{n}\sigma_k$
(a) Without DeepBlockIoTNet
$Pr_{[WD\_1]} = \frac{1}{4}\prod_{k=1}^{n}\phi_k \times \frac{1}{4}\prod_{k=1}^{n}\bar{\rho}_k$
$Pr_{[WD\_4]} = \frac{1}{4}\prod_{k=1}^{n}\phi_k \times \frac{1}{4}\prod_{k=1}^{v}\bar{\rho}_k$
Edge nodes (Learning agents)
Blockchain network
Blockchain Server
Edge nodes (Processing agents)
$Pr_{[WD\_2]} = \frac{1}{4}\prod_{k=1}^{v}\bar{\sigma}_k \times \frac{1}{4}\prod_{k=1}^{n}\phi_k$
Local models upload
Verified local model updates
List verified local model updates
Global models download
Block generation and mining
$Pr_{[WD\_3]} = \frac{1}{4}\prod_{k=1}^{v}\bar{\sigma}_k \times \frac{1}{4}\prod_{k=1}^{n}\phi_k$
(b) With DeepBlockIoTNet

Fig. 5: Security and privacy analysis of DeepBlockIoTNet

As illustrated in the results of the simulation in Fig. 4d, the probabilities in both "With DeepBlockIoTNet" and "Without DeepBlockIoTNet" cases decrease with an increasing number of edge nodes to be hacked. Note, however, that "With DeepBlockIoTNet" always obtained the lower value of the overall probability of attacks compared to "Without DeepBlockIoTNet". It could be due to the blockchain technology employed in the "With DeepBlockIoTNet" provides extra probability values (i.e., ϕ_k: the probability of stealing of the key's information for each node k by attackers) and the attacker needs to hack a large number of transmission mediums (i.e., $v = \theta \cdot n(n-1)/2$) for successful attacks. Since the value of ϕ_k in decimal and $(v > n)$, the decimal multiplication of ϕ_k to each element and larger value of v in equation 11 will jointly result in the lower value of $Pr_{[WD]}$ in comparison with $Pr_{[WoD]}$. The lower value of overall probability "With DeepBlockIoTNet" compared to "Without DeepBlockIoTNet", demonstrates that "With DeepBlockIoTNet" have less possibility to be attacked than "Without DeepBlockIoTNet" which satisfies the higher security requirement.

VI. Conclusion

In this paper, DeepBlockIoTNet, a secure DL approach with blockchain for the IoT network, contributes to the field of DL for big data analysis in next-generation CPS in three ways. First, a distributed DL approach was presented to deploy the DL operation at the edge layer (e.g., edge intelligence) and mitigate the challenges in the DL at the cloud layer (e.g., cloud intelligence). Second, the distributed DL was configured in a blockchain environment to mitigate the three major challenges of edge intelligence, specifically centralized control, security and privacy, and low accuracy and provide a secure, decentralized DL approach. Finally, DeepBlockIoTNet was experimentally evaluated with an object detection application to demonstrate the practicability and the Quality of Service (QoS) of the proposed DeepBlockIoTNet in practical applications. However, DeepBlockIoTNet has certain challenges in the context of cross-communication between multiple domains in IoT wherein sensors vary, and different sensors collect different kinds of knowledge, which result in different DL models for different domains. To address these challenges, DeepBlockIoTNet can be enhanced by using transfer learning and sensor fusion techniques.

References

[1] Kim, N. Y., Rathore, S., Ryu, J. H., Park, J. H., & Park, J. H. (2018). A Survey on Cyber Physical System Security for IoT: Issues, Challenges, Threats, Solutions. *Journal of Information Processing Systems*, *14*(6).

[2] Vengadeswaran & Balasundaram. (2019): CORE-An optimal data placement strategy in Hadoop for data intensive applications based on cohesion relation. *Computer Systems Science and Engineering,* Vol. 34, No. 1, pp. 47-60.

[3] C. S. Wickramasinghe, D. L. Marino, K. Amarasinghe, and M. Manic, "Generalization of Deep Learning for Cyber-Physical System Security: A Survey," *IECON 2018 - 44th Annual Conference of the IEEE Industrial Electronics Society*, 2018.

[4] Truong, M. T. N., & Kim, S. (2019). A tracking-by-detection system for pedestrian tracking using deep learning technique and color information. Journal of Information Processing Systems, 15(4), 1017-1028.

[5] Garg, S., Kaur, K., Kumar, N., Kaddoum, G., Zomaya, A. Y., & Ranjan, R. (2019). A hybrid deep learning-based model for anomaly detection in cloud datacenter networks. *IEEE Transactions on Network and Service Management*, *16*(3), 924-935.

[6] Q. Zhang, L.T. Yang, z. Chen, P. Li, and M.J. Deen, "Privacy-preserving double-projection deep computation model with crowdsourcing on cloud for big data feature learning", *IEEE Internet of Things Journal,* vol. 5, no. 4, pp. 2896-2903, 2018.

[7] Deng, Z., Ren, Y., Liu, Y., Yin, X., Shen, Z., & Kim, H. J. (2019). Blockchain-based trusted electronic records preservation in cloud storage. *Comput. Mater. Continua*, *58*(1), 135-151.

[8] M.Chen, Y. Hao, K. Lin, Z. Yuan, and L. Hu, "Label-less learning for traffic control in an edge network", *IEEE Network,* vol. 32, no. 6, pp. 8-14, 2018.

[9] H. Li, K. Ota, and M. Dong, "Learning IoT in edge: deep learning for the internet of things with edge computing", *IEEE Network,* vol. 32, no. 1, pp. 96-101, 2018.

[10] Garg, S., Kaur, K., Kumar, N., & Rodrigues, J. J. (2019). Hybrid deep-learning-based anomaly detection scheme for suspicious flow detection in SDN: A social multimedia perspective. *IEEE Transactions on Multimedia*, *21*(3), 566-578.

[11] *Health insurance portability and accountability act.* [Online]. Available: https://www.hhs.gov/hipaa/index.html.

[12] R. Shokri, and V. Shmatikov, "Privacy-preserving deep learning," *in Allerton Conference on Communication, Control and Computing*, 2015, pp. 909–910.

[13] A. Sergeev, and M. Del Balso, "Horovod: fast and easy distributed deep learning in TensorFlow", *arXiv preprint arXiv*, pp. 1802.05799, Feb. 2018.

[14] Kaur, K., Garg, S., Kaddoum, G., Gagnon, F., & Ahmed, S. H. (2019, May). Blockchain-based lightweight authentication mechanism for vehicular fog infrastructure. In *2019 IEEE International Conference on Communications Workshops (ICC Workshops)* (pp. 1-6). IEEE.

[15] Bordel, B., Alcarria, R., Martin, D., & Sanchez-Picot, A. (2019). Trust Provision in the Internet of Things Using Transversal Blockchain Networks. *INTELLIGENT AUTOMATION AND SOFT COMPUTING*, *25*(1), 155-170.

[16] S. Biswas, K. Sharif, F. Li, B. Nour, and Y. Wang, "A Scalable Blockchain Framework for Secure Transactions in IoT," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4650–4659, 2019

[17] R. Yang, F. R. Yu, P. Si, Z. Yang, and Y. Zhang, "Integrated Blockchain and Edge Computing Systems: A Survey, Some Research Issues and Challenges," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 2, pp. 1508–1532, 2019

[18] Nyamtiga, Sicato, Rathore, Sung, and Park, "Blockchain-Based Secure Storage Management with Edge Computing for IoT," *Electronics*, vol. 8, no. 8, p. 828, 2019

[19] S. Rathore, B. W. Kwon, and J. H. Park, "BlockSecIoTNet: Blockchain-based decentralized security architecture for IoT network," *Journal of Network and Computer Applications*, vol. 143, pp. 167–177, 2019

[20] Hung, C. W., Mao, W. L., & Huang, H. Y. (2019). Modified PSO Algorithm on Recurrent Fuzzy Neural Network for System Identification. *Intelligent Automation And Soft Computing*, *25*(2), 329-341

[21] Salakhutdinov Ruslan, and E. Hinton Geoffrey, "Deep Boltzmann machines on Artificial Intelligence and Statistics", *in Proc. of The 12th Int. Conf.*, 2009, PMLR 5, pp. 448-455.

[22] Oh, B. D., Song, H. J., Kim, J. D., Park, C. Y., & Kim, Y. S. (2019). Predicting Concentration of PM10 Using Optimal Parameters of Deep Neural Network. *Intelligent Automation and Soft Computing*, *25*(2), 343-350.

[23] C. Hardy, E. Merrer Le, and B. Sericola, "Distributed deep learning on edge-devices: feasibility via adaptive compression", *in Proc. 2017 IEEE 16th Int. Symposium on Network Computing and Applications (NCA)*, pp. 1-8, Oct. 2017.

[24] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, and A.Y. Ng, "Large scale distributed deep networks", *Advances in neural information processing systems*, pp. 1223-1231, 2012.

[25] J. S. Weng, J. Weng, M. Li, Y. Zhang, W. Luo, "DeepChain: Auditable and Privacy-Preserving Deep Learning with Blockchain-based Incentive," *IACR Cryptology ePrint Archive*, vol. 2018, p. 679, 2018

[26] S. Rathore, Y. Pan, and J. H. Park, "BlockDeepNet: A Blockchain-Based Secure Deep Learning for IoT Network," *Sustainability*, vol. 11, no. 14, p. 3974, 2019

[27] M. Pilkington, "Blockchain technology: principles and applications," *Research Handbook on Digital Transformations*, ed. by F. Xavier Olleros and Majlinda Zhegu. Edward Elgar, 2016.

[28] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, pp. 1-32, 2014

[29] Liu, M., Cheng, L., Qian, K., Wang, J., Wang, J., & Liu, Y. (2020). Indoor acoustic localization: a survey. Human-centric Computing and Information Sciences, 10(1), 2.

[30] Park, J., & Park, K. (2020). Construction of a remote monitoring system in smart dust environment. Journal of Information Processing Systems, 16(3), 733-741.

[31] Garg, S., Kaur, K., Kaddoum, G., Ahmed, S. H., & Jayakody, D. N. K. (2019). SDN-based secure and privacy-preserving scheme for vehicular networks: A 5G perspective. *IEEE Transactions on Vehicular Technology*, *68*(9), 8421-8434.

[32] Park, J. H., Salim, M. M., Jo, J. H., Sicato, J. C. S., Rathore, S., & Park, J. H. (2019). CIoT-Net: a scalable cognitive IoT based smart city network architecture. Human-centric Computing and Information Sciences, 9(1), 29.

[33] COCO dataset. [Online]. Available: http://cocodataset.org/#download.

[34] Zhang, Y., Huang, M., Wang, H., Feng, W., Cheng, J., & Zhou, H. (2019). A co-verification interface design for high-assurance CPS, Computers, Materials & Continua, Vol. 58, No. 1, pp. 287-306.

[35] Tian, W., Ji, X., Liu, W., Liu, G., Lin, R., Zhai, J., & Dai, Y. (2019). Defense strategies against network attacks in cyber-physical systems with analysis cost constraint based on honeypot game model. *Comput. Mater. Continua*, *60*(1), 193-211.

Dr. Shailendra Rathore received Ph.D. degrees in the Graduate School of Computer Science and Engineering at Seoul National University of Science and Technology, Seoul, South Korea (June 2020). Currently, Dr. Rathore is working as a researcher in the Ubiquitous Computing Security (UCS) Lab lead by Prof. Jong Hyuk Park. His broad research interest includes Information and Cyber Security, IoT, Artificial Intelligence, Blockchain, and Deep Learning.

Dr. Jong Hyuk Park received Ph.D. degrees in Korea University, Korea and Waseda University, Japan. He is now a professor at the Department of Computer Science and Engineering, Seoul National University of Science and Technology, Korea. Dr. Park has published about 300 research papers in international journals and conferences. He is editor-in-chief of Human-centric Computing and Information Sciences (HCIS) by Springer, The Journal of Information Processing Systems (JIPS) by KIPS. His research interests include IoT, Information Security, Smart City, Blockchain, etc.