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/TETC.2020.2983007, IEEE Transactions on Emerging Topics in Computing

A Blockchain-Powered Decentralized and Secure Computing Paradigm

Gihan J. Mendis*, Yifu Wu*, Jin Wei*, Moein Sabounchi*, and Rigoberto Roche'†
*Purdue University, West Lafayette, IN 47907
†NASA Glenn Research Center, Cleveland, OH 44135

***Abstract*—Thanks to the advances in machine learning, data-driven analysis tools have become valuable solutions for various applications. However, there still remain essential challenges to develop effective data-driven methods because of the need to acquire a large amount of data and to have sufficient computing power to handle the data. In many instances these challenges are addressed by relying on a cloud computing vendor. However, although commercial cloud vendors provide valuable platforms for data analytics, they can suffer from a lack of transparency, security, and privacy-preservation. Furthermore, reliance on cloud servers prevents applying big data analytics in environments where the computing power is distributed. To address these challenges, a decentralized, secure, and privacy-preserving computing paradigm is proposed to enable an asynchronized cooperative computing process amongst distributed and untrustworthy computing nodes that may have limited computing power and computing intelligence. This paradigm is designed by exploring blockchain, decentralized learning, and homomorphic encryption techniques. The performance of the proposed paradigm is evaluated via different scenarios in the simulation section.**

***Index Terms*—Blockchain, Decentralized and Secure Learning, Machine Learning, Privacy, Security, Homomorphic Encryption**

I. Introduction

Due to the advances in sensing and computing, data-driven methods, such as machine learning techniques [1], have become very promising solutions for different applications [2]–[4]. However, there still remain some essential challenges in implementing these techniques practically, including: (1) acquisition of large amount of data, and (2) requirement of sufficient computing power, which enforces the reliance on a cloud computing vendor. Although the cloud servers provide valuable platforms for big data analytics, there are still several essential challenges in adopting the commercial cloud vendors: (1) transparency, (2) security, and (3) privacy [5]. Furthermore, the reliance on cloud servers also limits the potential of applying big data analytics for the environment where the computing power is scattered. Therefore, it is meaningful to develop a reliable infrastructure in which end users are incentivized to contribute to machine learning tasks in a privacy-preserving manner. To achieve this goal, in this paper, we develop a decentralized and secure computing infrastructure that enables an effective and privacy-preserving collaboration between the available end users that are called computing nodes. These computing nodes can have restricted computing power and limited computing intelligence. Additionally, they can be distributed and untrustworthy to each other.

Several techniques have been proposed to achieve decentralized and privacy-preserving computing. In [6], Shokri *et al.* proposed a privacy-preserving deep-learning mechanism with secure multi-party computations to update the single initial deep learning model. The deep learning model is centralized while multiple parties contribute to the model training in a manner that guaranties the privacy-preservation. Federated learning introduced in [7]–[10] is a distributed machine learning method that enables model training on decentralized data. In this method, multiple copies of the central model are available in distributed computing devices for training, which eliminates the single point of failure. Both of these two methods must be managed by one centralized authoritative controlling agent, which may raise security and privacy concerns. To address these issues, in our work we develop a decentralized and secure computing paradigm, which does not have any centralized authoritative controlling agents, by exploiting blockchain, machine learning, and homomorphic encryption technologies. Blockchain is an emerging technology, which can be considered as an immutable and decentralized digital ledger [11], [12]. A blockchain is a growing list of records, called blocks, which are linked via cryptography. Each block contains the hash value of the previous block, the transaction data, and the timestamp. Due to the inherent cryptographic chaining, if a malicious party tries to manipulate certain transaction data, the hash value of the block containing the transaction and those of all the subsequent blocks will change, which can be easily detected. Therefore, generally speaking, blockchain technology provides a very promising solution for integrity security. Amongst various existing blockchain platforms, Ethereum and Bitcoin are two of the most widely adopted ones [11]–[13]. Compared with Bitcoin, Ethereum platform provides a trustful automation of programs via smart contracts that run on virtual machines. In our work, we exploit Ethereum blockchain to execute the secure and privacy-preserving decentralized computing functionalities automatically. Furthermore, our computing paradigm enables the decentralized and cooperative learning via an effective learning-model fusion mechanism. Fusing multiple learning models is an active area of research and fusion strategies in literature can be divided into two main categories: (1) *Late fusion* that comprises predicting the labels based on the labels given by each learning model to be fused and (2) *Early fusion* that takes the feature vectors given by the individual learning models as the inputs and learns a classifier on top of them. Although *late fusion* requires lower computational cost compared with *early fusion* in many practical applications [14]–[16], *early fusion* can enable an optimal way to integrate learned models [17]. In this work, our learning-model fusion mechanism belongs to the type of *early fusion*. Additionally, we consider two strategies for designing the fusion mechanism: (1) using a fully connected structure with a single hidden-layer to map the concatenated features to the labels, and (2) implementing a gradual fusion strategy to explore the uniqueness of the individual learning models and the correlation amongst the learning models. To further enhance security and privacy-preservation, we design an encryption interface with a zero-knowledge proof protocol by exploiting homomorphic encryption (HE) technology, which enables evaluating the performance of the contributed learning models without revealing the sensitive details of the learning models. HE technology is one form of encryption that allows the computation operations to be directly implemented in a cipherspace and achieves an encrypted results that, when decrypted, match the results of the operations as if they were performed on a plainspace [18]. The existing HE technologies can be generally classified into three main groups: (1) fully HE schemes, (2) partially HE schemes, and (3) somewhat HE schemes [19]–[24]. Considering the fact that somewhat HE schemes support more operations compared with partially HE schemes and require less computation power compared with fully HE schemes, we exploit Integer-Vector HE scheme [24], which is a somewhat HE scheme, to develop the encryption interface in our computing

paradigm. The authors would like to claim that the technologies presented in this paper has been include in a provisional patent [25].

The next section reviews the previous related works. Section II discusses the existing work related to our proposed computing paradigm. Section III describes the problem setting for our work. Section IV presents our proposed blockchain-powered decentralized and secure computing paradigm mechanism. The details of the implementation is illustrated in Section V. Section VI provides security analysis of our work. Simulation results and the conclusions are shown in Sections VII and VIII, respectively.

II. Related Work

A. Distributed Computing Methods

There still remain essential challenges to apply deep learning directly to distributed computing systems. To address the challenges, several distributed computation strategies have been developed [7]–[9], [26]–[28]. The methods in [26]–[28] were designed based on MapReduce [29], where the database is partitioned and distributed to each computing node that has a replicated version of the learning model. During the model training, each computing node needs to update the hyperparameters of their model based on the updated learning models of other participants. The authors in [7]–[9] developed federated learning to periodically update the model weights between each computing node after a certain number of local epochs, which is based on stochastic Gradient descent technique. In these established methods, the distributed computing nodes are effectively leveraged. However, the dramatically increased communication complexity becomes a critical issue. For example, the time complexity, especially the communication complexity, of federated learning is linear to the number of training epochs. To mitigate this critical issue, in this paper, we propose a fusion model based on transfer learning [30], which minimizes the communication overhead to a constant number. By doing so, the time complexity of our model remains constant even the number of training epochs is large. Furthermore, our proposed method enhances the security and privacy of the distributed learning by utilizing homomorphic encryption techniques. A comparison between our model and other established distributed computing methods is shown in Table I.

B. Management of Distributed Computing Resources

To release the pressure of centralized cloud computing, different distributed computing systems have been proposed to support edge and fog computing. However, there still remain challenges to incentivize and manage the participation of the distributed computing resources. Several research results have been established to address these challenges. In [31], the authors provided three feasible mechanisms with different efficiency and truthfulness to balance the benefits between mobile devices and cloudlets. In [32], the authors targeted at improving the strategy of the distributed computation system by adaptively calculating the utilities of computation resources in multiple dimensions. Zhang *et al.* [33] developed a framework for online cloud auctions that analyzes the diverse inquiries of users. The proposed allocation rule is to maximize the utilization of whole computation resources. Additionally, in [34], a three-stage auction mechanism was proposed to efficiently allocate distributed computing resources, such as mobile devices, to other nearby mobile users. These established mechanisms are effective in leveraging the distributed computing resources. However, all these methods were designed by assuming there is a trustworthy and centralized third-party platform. Recently, blockchain technologies [11], [12] provide a promising solution to relax this assumption, which is not always practical. Buyya *et al.* [35] applied blockchain to manage and protect distributed data collection in Internet-of-Thing (IoT) system. In [36], the authors proposed a blockchain-based decentralized cloud task scheduler that utilizes a novel consensus algorithm called "proof-of-schedule". In this work, the authors have also experimentally validated the proposed method using a cloud simulator. Xiong *et al.* proposed an edge computing enabled mobile blockchain network with a prototype and a pricing scheme in [37]. The blockchain network enables IoT or mobile devices to utilize computing services from an edge computing service provider. The authors in [38] utilized a private blockchain, Hyperledge, to manage a decentralized edge computing of IoT system, in which the computation is centralized on local devices. In this paper, we explore blockchain techniques from another aspect, which is to enable decentralized and secure cooperation amongst distributed computing nodes. The comparison analysis between our method and the one proposed in [35] is shown in Table I.

III. Problem Setting

Two of the main factors for the thriving of machine learning are: (1) the availability of sufficient data with generalized distributions, commonly known as big data, and (2) the availability of adequate computing power to process the big data, mainly in the form of large-scale GPU clusters. Because of this, most profitable parties in the field of machine learning are large organizations that possess both valuable big data and sufficient computing power. As illustrated in Fig. 1(a), these large organizations collect data from computing contributors to advance the capabilities of their machine learning techniques. One of the essential challenges for creating a large-scale dataset via data acquisition, from multiple parties, is the concern of the potential privacy violations of the computing contributors. Additionally, collecting tremendous raw data from individual computing contributors may result in a huge demand for communication bandwidth and a dramatically increased attack surface. Furthermore, a large amount of computational power is required by the central server to process the collected big data.

One solution is the implementation of distributed computing architectures [6]–[10], [26]–[28]. As shown in Fig. 1(b), in distributed computing, rather than collecting and processing data in a single central server, data processing is distributed partially to the individual computing contributors. By doing so, the distributed computing is implemented in such a way, that the contributors process their local data by training the given machine learning models or their own machine learning models and then share the trained model with a central controlling agent. Since the local raw data are not shared directly, the data privacy is enhanced in this architecture. Additionally, the machine learning models are trained in a distributed manner with smaller sets of data, and thus the computing power required by the individual computing contributors is much lower, compared with that of a central server. However, in this solution, the distributed computing is fully controlled by an authoritative agent in a centralized manner. It relies on the central authority to coordinate the activities of each entity in the system. Therefore, it is required that the computing contributors trust the central controlling agent, which may raise security and privacy concerns.

To mitigate this issue, we develop a blockchain-powered decentralized and secure computing paradigm, as shown in Fig. 1(c), to realize two main objectives. First, we target at tackling the potential issues of limited availability of computing power and data by developing a decentralized and secure machine learning paradigm. In our developed paradigm, the distributed computing contributors, which can be untrustworthy, have full controls of their own deep learning models and private data. Additionally, the individual contributors are able to participate or leave the computing architecture

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/TETC.2020.2983007, IEEE Transactions on Emerging Topics in Computing

TABLE I
COMPARISON BETWEEN THE PERFORMANCE OF OUR PROPOSED METHOD AND THOSE OF OTHER EXISTING RELATED METHODS

Name	Autonomy in Decentralized System	Anonymity	Data Integrity	Data Privacy	Data Identification	Edge Computing	Distributed Model Inference	Distributed Model Training
Our method	✓	✓	✓	✓	✓	✓	✓	✓
QoS-aware IoT [35]	✓	✓	✓	✓	✗	✗	✗	✗
FogBus [28]	✗	✗	✓	✓	✗	✓	✓	✗
EdgeLens [27]	✗	✗	✗	✗	✗	✓	✓	✗
Federated learning [9]	✗	✗	✗	✗	✗	✓	✓	✓
Auction-based Mobile Cloud [39]	✗	✗	✗	✗	✗	✓	✗	✗
Blockchain Scheduling [36]	✓	✓	✓	✗	✗	✗	✗	✗
Edge Computing Mobile Blockchain [37]	✓	✓	✓	✗	✗	✓	✗	✗
Blockchain IoT Control [38]	✓	✗	✓	✗	✗	✓	✗	✗

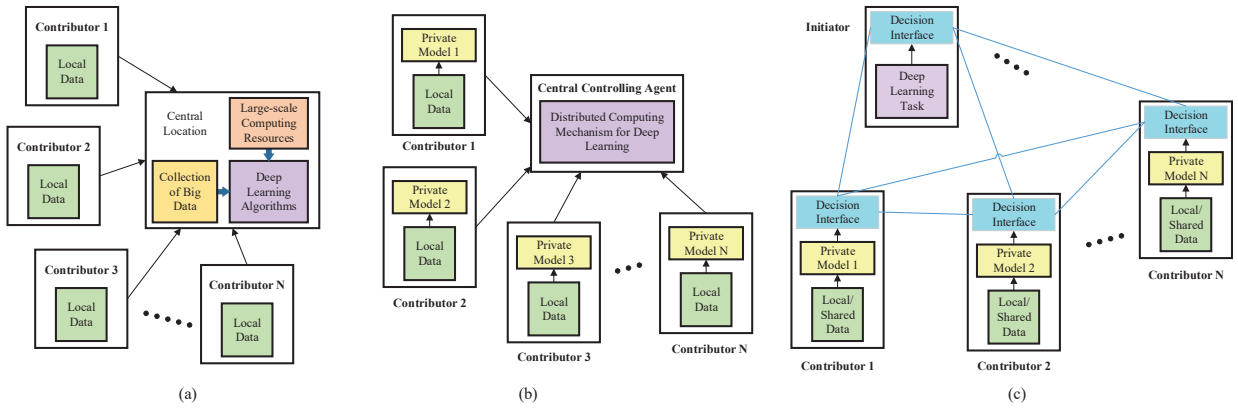


Fig. 1. (a) Centralized machine learning architecture where data are collected by a centralized server that has high processing and storage capability; (b) Distributed computing architecture where the training process is fully supervised by a central controlling agent; (c) Autonomous cooperative and decentralized machine learning architecture with no central agents facilitated by blockchain techniques.

at any time, without disturbing the functionality and efficiency of the overall learning process. Second, we focus on enhancing the security and privacy of the decentralized and secure machine learning even when there exist untrustworthy contributors. In this objective, we mainly consider computation integrity and data confidentiality. For computation integrity, we consider two concrete threats: (1) Byzantine fault that is a type of fault presenting different symptoms to different observers [40] and (2) lazy or dishonest contributors that are the computing entities intentionally providing inefficient or incorrect results to gain economic benefits [41]. To address Byzantine fault, our computing paradigm exploits blockchain technology with Proof-of-Work (PoW) consensus protocol. To mitigate the impact of lazy or dishonest contributors, our computing paradigm comprises verification contributors with a majority voting mechanism for evaluating the quantitative contributions of the claimed private models. For data confidentiality, we consider one concrete threat: honest-but-curious contributors that are the computing entities trying to discover the sensitive and private information from the data [42]. We design our computing paradigm to defend against this threat from two perspectives: (1) realizing decentralized and secure computing via the sharing of learning models rather than raw data that may contain sensitive and private information, and (2) further enhancing the confidentiality of learning-model sharing by exploiting homomorphic encryption techniques. The authors would like to mention that the detailed definition of the threat models for Byzantine, lazy or dishonest contributors, and honest-but-curious contributors will be described in Section VI. The security analysis of our proposed computing paradigm with respect to these three threat models will also be provided in that section.

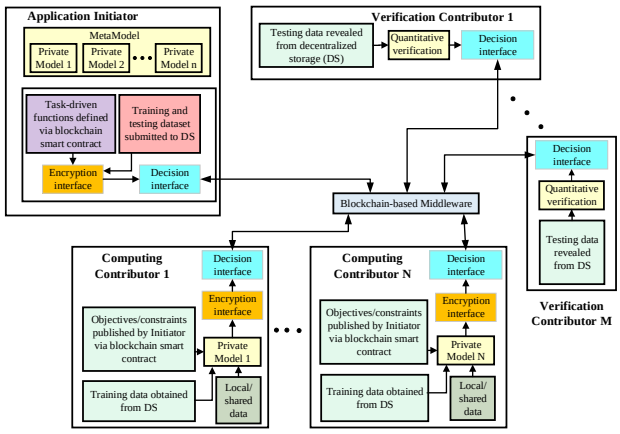


Fig. 2. Overview of our blockchain-powered decentralized and secure computing paradigm.

IV. PROPOSED BLOCKCHAIN-POWERED DECENTRALIZED AND SECURE COMPUTING PARADIGM

The detailed implementation of our proposed blockchain-powered decentralized and secure computing paradigm is illustrated in Fig. 2. As shown in Fig. 2, our proposed paradigm is designed to enable the effective cooperation between the available distributed computing nodes to accomplish a data-driven task that requires high computing power and large dataset. The individual computing nodes can participate in the computing paradigm by playing one of the three roles: (1) application initiators, (2) computing contributors, and (3) verification contributors. As described in Fig. 3(a), if the computing node acts as an application initiator, it announces the data-driven applications and is responsible for defining the computing tasks, such

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/TETC.2020.2983007, IEEE Transactions on Emerging Topics in Computing

as the objectives, constraints, and the suggestions on the computing model, via blockchain smart contract. Additionally, the application initiator also provides the verification contributors with a sample set of data to evaluate the performance of the learning models contributed by the computing contributors. If it is necessary, the application initiator may also provide the computing contributors, which have computing power and computing intelligence, with the dataset to conduct the local training. The sharing of the dataset is realized via the decentralized storage (DS) such as The Interplanetary File System (IPFS) [43]. As shown in Fig. 3(b), if the computing node acts as a computing contributor, it trains the deep learning models locally for a given data-driven task by using the available local data or the data shared by the associated application initiator. After training the local machine learning model successfully according to the criteria defined by the application initiator via smart contract, such as the accuracy above 90 %, the computing contributor announces the completeness of the training via the blockchain-based middleware and shares the achieved learning model to the randomly selected verification contributors via the DS such as IPFS. Then the available verification contributors are passively and randomly selected to provide the hardware resources and verify the contributions of the locally trained learning models, which are claimed by the computing contributors, in a decentralized manner. As illustrated in Fig. 3(c), the quantitative verification is conducted according to the criteria defined by the associated application initiator via smart contract, such as whether the accuracy can be improved after fusing the claimed model. The majority voting amongst the passively and randomly selected verification contributors is used to determine the contribution of the corresponding locally trained learning models. The application initiator and all the selected verification contributors are informed about the majority voting result. If this result is positive, the transaction of the verified locally-trained machine-learning model, also called private model, is established between the application initiator and the associated computing contributor, in which the computing contributor receives the financial compensation and the application initiator obtains the access to the private model in IPFS. Additionally, the verification contributors involved in the task also get compensated from initiators for their effort. After the time window, which is assigned to the given data-driven task, ends, the application initiator fuses all the received verified private models to achieve the MetaModel that will be applied to address the data-driven task. The architecture of our proposed computing paradigm is considered asynchronous and adaptive since the computing and verification contributors can leave or join the task at their convenience.

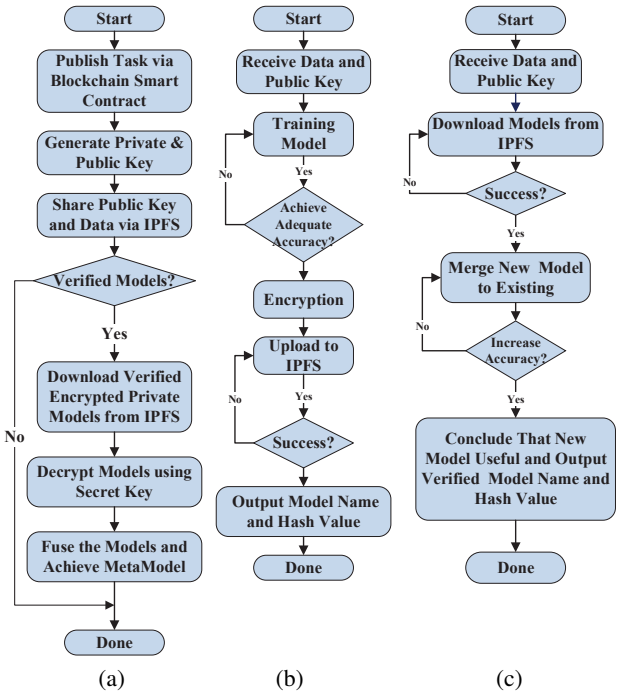


Fig. 3. The schematic diagrams showing workflows for (a) application initiators, (b) computing contributors, and (c) verification contributors.

V. Implementation of Proposed Computing Paradigm

In this section, we describe the implementation of our proposed blockchain-powered decentralized and secure computing paradigm in details. The implementation of our proposed computing paradigm mainly consists of computing layer, blockchain-enabled middleware, and networking layer. As shown in Fig. 2, the computing layer is designed to conduct decentralized and secure computing, which comprises decentralized storage system and homomorphic encryption to enhance the security and privacy-preservation of our computing paradigm. Furthermore, our blockchain-enabled middleware is developed to realize the secure peer-to-peer communications between the computing nodes including the transactions of the private models and the data sharing.

A. Decentralized Storage System

As shown in Fig. 2, decentralized storage (DS) system is one essential component of the computing layer to secure the sharing of data and private machine-learning models. DS system makes the access these data and learning models more secure, affordable, and faster. Firstly, expensive centralized data servers are no longer needed in a DS system because every peer can cache the data required by other peers. Secondly, optimized downloading from multiple local peers provides higher traffic throughput than that from a remote centralized data server. In addition, as shown in Fig. 3, after the data or private models are stored in the DS system, the unique hashes characterizing the fingerprint of these cyber assets are generated and shared, which ensures the integrity and the authorization of the sharing process.

B. Homomorphic Encryption Interface

As shown in Fig. 4, the encryption interface is designed to enhance the security and enable the privacy-preservation of our computing paradigm. In this work, the encryption interface is developed by exploiting Integer Vector Homomorphic Encryption (HE) scheme. Figure 4 illustrates the overall mechanism of the encryption interface, which mainly consists of eight steps. Step 1: Encryption interface client of the application initiator generates the public key $\mathbf{M}$ and the secret key $\mathbf{S}'$ according to Integer Vector HE scheme, which is illustrated in Fig. 3(a). Step 2: The generated public key $\mathbf{M}$ is shared amongst the active computing and verification contributors, which is illustrated in Figs. 3(a)-(c). Steps 3 and 4: Computing contributors apply the received public key $\mathbf{M}$ to encrypt the machine learning models achieved locally and share the encrypted private models with the passively and randomly selected verification contributors for quantitative verification, which is illustrated in Fig. 3(b). Step 5: After receiving the encrypted private models, the verification contributors verify the performance of the models by conducting the quantitative verification in cipherspace with the public key $\mathbf{M}$. In the quantitative verification, the verification contributors fuse each of the received private models with the existing learning model in cipherspace and conclude that a private model is valuable if the overall accuracy increases after model fusion. Step 6: The majority voting amongst all the associated verification contributors is used to determine the contribution of each private model. The application

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/TETC.2020.2983007, IEEE Transactions on Emerging Topics in Computing

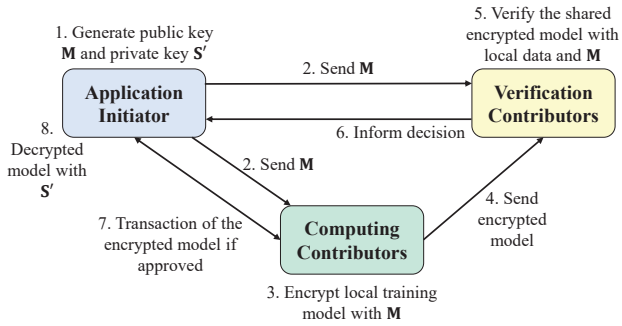


Fig. 4. The illustration of the mechanism of our encryption interface.

initiator is informed about the majority voting conclusion. Step 7: If the majority conclusion is positive, the transaction of the encrypted private model between the associated computing contributor and the application initiator is established. At last, the application initiator decrypts the shared model with the private key $\mathbf{S}'$.

1) Integer-Vector Homomorphic Encryption: In our work, the integer-vector homomorphic encryption scheme, which supports three fundamental operations: addition, linear transformation, and weighted inner product, is exploited to develop the encryption interface. Letting $x \in \mathbb{Z}^m$ be the plaintext vector, $\mathbf{S} \in \mathbb{Z}^{m \times n}$ be the secret key, and $c \in \mathbb{Z}^n$ be the ciphertext vector, the encryption can be formulated as:

$$\mathbf{Sc} = w\mathbf{x} + \mathbf{e}, \tag{1}$$

where$\mathbf{e}$ is a randomization term introduced to enable the encryption, which has elements smaller than w, and w is a large integer that controls the appropriate ratio between the plaintext and the introduced randomization.

Given the secret key $\mathbf{S}$, the decryption can be performed as:

$$\mathbf{x} = \lceil \frac{\mathbf{Sc}}{w} \rfloor. \tag{2}$$

In this homomorphic encryption scheme, *key switching* method is applied to convert the ciphertext in one cipherspace to another without decryption. This convert is realized via a public key $\mathbf{M}$ that is calculated as follows:

$$\mathbf{M} = \begin{bmatrix} \mathbf{S}^* - \mathbf{TA} + \mathbf{E} \\ \mathbf{A} \end{bmatrix} \tag{3}$$

where $\mathbf{S}$ and $\mathbf{c}$ are the original private key and ciphertext, respectively, $\mathbf{S}^*$ is an intermediate key satisfying $\mathbf{S}^*\mathbf{c}^* = \mathbf{Sc}$, $\mathbf{c}^*$ is the representation of $\mathbf{c}$ with each digit of $\mathbf{c}$, c_i represented as a l-bit binary number, and $\mathbf{A}$ and $\mathbf{E}$ are random and bounded matrices, respectively. A scalar l is selected to be large enough such that $|\mathbf{c}| < 2^l$, which determines the maximum value of $\mathbf{c}$ represented with l bits. Letting $\mathbf{b}_i$ be the bit representation of the value c_i, we can obtain $\mathbf{c}^*$ as follows:

$$\mathbf{c}^* = [\mathbf{b}_i, \mathbf{b}_{i+1}, ..., \mathbf{b}_n] \tag{4}$$

where n is the length of vector $\mathbf{c}$. Similarly, $\mathbf{S}^*$ can be obtained as follows:

$$\mathbf{B}_{ij} = \left[2^{l-1}S_{ij}, ..., 2S_{ij}, S_{ij}\right] \tag{5}$$

where $\mathbf{B}_{ij}$ is the sub-vector of $\mathbf{S}^*$ that corresponds to the element S_{ij}.

Additionally, since the initial private key $\mathbf{S}$ is an identity matrix of the dimension $n \times n$, the original ciphertext $\mathbf{c}$ is equal to the original plaintext $\mathbf{x}$. Let $\mathbf{S}' = [\mathbf{I}, \mathbf{T}]$, where $\mathbf{I}$ is an identity matrix and $\mathbf{T}$ is a desired secret vector. By using the public key $\mathbf{M}$ defined in Eq. (3), the ciphertext corresponding to the desired private key $\mathbf{S}'$ can be calculated as:

$$\mathbf{c}' = \mathbf{Mc}^* \tag{6}$$

where $\mathbf{M}$ is a $(n+1) \times nl$-dimension matrix. Therefore, the resulting ciphertext $\mathbf{c}'$ is an integer vector with length $n+1$. $\mathbf{M}$ is formulated such that $\mathbf{M}$ satisfies following equality:

$$\mathbf{S}'\mathbf{M} = \mathbf{S}^* + \mathbf{E} \tag{7}$$

2) Implementation of Artificial Neural Networks-based Machine Learning Model in Cipherspace: As shown in Fig. 4, the essential component of our homomorphic encryption is to implement the artificial neural network (ANN)-based machine learning model in cipherspace by using integer-vector homomorphic encryption (IVHE) scheme. ANN implementation mainly comprises of summation, vector addition, vector multiplication with a scalar, vector dot product, pooling operations, convolution operations, and nonlinear activation functions. Most of these operations are supported by the IVHE scheme, except pooling operations, convolution operations, and nonlinear activation function realizations. To enable the implementation of pooling operations in cipherspace, in this work, we assume the computing contributors adopt the average pooling or summation pooling in training their local machine-learning models. Under this assumption, the pooling operations can be realized via the summation followed by a division by an integer, which is supported by the IVHE. The convolution operations can be implemented in cipherspace by calculating the vector multiplication operations. Additionally, we mainly consider two types of activation functions: sigmoid function, $\sigma(x) = 1/(1+e^{-x})$, and ReLU function, $ReLU(x) = max(0, x)$. To implement the sigmoid functions in cipherspace, we leverage the Taylor series expansion to achieve the k-th order polynomial approximation of the sigmoid function. For example, if $k = 3$, the polynomial approximation is $\bar{\sigma} = \frac{1}{2} + \frac{x}{4} - \frac{x^3}{348}$ that is supported by the IVHE scheme. To enable the implementation of ReLU function, which is a piecewise linear operation with discontinuity at $x = 0$, in cipherspace, we currently constrain the secret and public keys to contain non-negative elements only. By doing so, we are able to ensure that there is no sign change while encrypting. The ReLU function is executed in cipherspace via random dropout.

There still remain two challenges of implementing the ANN-based machine learning model in cipherspace. First, most weights and input data for ANNs are floating-point values that cannot directly be supported by the adopted homomorphic encryption scheme. Additionally, implementing the average pooling and calculating the polynomial approximation of sigmoid function require the multiplications with floating-point numbers, which is also not supported by our adopted homomorphic encryption scheme. To address this issue, we introduce predetermined and unified scaling factors to convert the floating-point values to integers. A final propagated scaling factor is used to scale down the output values of the ANNs to the original values. Second, as discussed in Section V-B1, integer-vector homomorphic encryption increases the length of ciphertext vector by 1 compared with that of the plaintext vector. This difference between the dimensions of the ciphertext and plaintext raises challenges in implementing the ANN operations that require consistency in dimensions such as feed-forward operations. To address this issue, we develop two encryption strategies.

a) Element-wise Encryption Strategy: The essential idea of our element-wise encryption strategy is to encrypt the matrices and vectors in an element-by-element manner. By doing so, the additional components introduced by the homomorphic encryption can be addressed in the third dimension, which ensures the consistency on the original dimensions. To illustrate our strategy, we use a fully-connected neural network (NN) as an example shown in Fig. 5(a). The details of the implementation of the fully-connected NN in cipherspace are illustrated in Fig. 5(b).

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/TETC.2020.2983007, IEEE Transactions on Emerging Topics in Computing

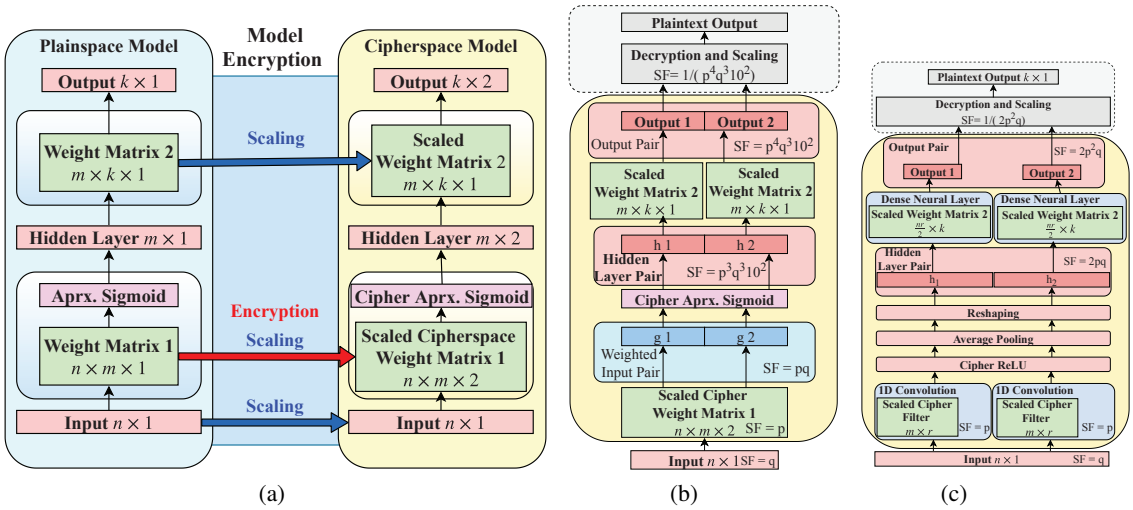


Fig. 5. The illustration of our element-wise encryption strategy: (a) using a fully-connected NN as an example, (b) detailing the execution of the fully-connected NN in cipherspace, and (c) using a CNN as an example.

As shown in Fig 5(a), the fully-connected ANN has an input vector with the length of n, a weight matrix, Weight Matrix 1, with the dimension of $n \times m$ resulting in a hidden layer of size m adopting sigmoid function as the activation function, a weight matrix, Weight Matrix 2, with the dimension of $m \times k$, and an output vector of length k. To enable the implantation of our element-wise encryption strategy, Weight Matrices 1 and 2 are represented with the dimensions of $n \times m \times 1$ and $n \times k \times 1$, respectively, and multiplied with a scaling factor p to ensure all the matrix elements to be integer. Another scaling factor q is introduced to convert the input elements to be integer. Considering the fact that, if Weight Matrix 1 is encrypted, the following structure of the fully-connected NN is meaningless for a malicious party, it is reasonable to focus on encrypting Weight Matrix 1 only to achieve a tradeoff between the high security and low computational complexity. As shown in Figs. 5(a) and (b), we consider each element of Weight Matrix 1 as a vector with the length of 1 in the 3rd dimension, and apply the element-wise encryption strategy, which results in a weight matrix with the dimension of $n \times m \times 2$ in cipherspace. The dot product operation between the scaled and encrypted Weight Matrix 1 and the scaled input vector is executed by using IVHE, which results in a pair of weighted inputs in cipherspace. By achieving a polynomial approximation of the sigmoid activation function, the hidden layer with the approximated sigmoid functions is encrypted via the homomorphic encryption resulting in a hidden layer pair in cipherspace. For each of the encrypted hidden layer in the pair, a dot product operation is performed with the scaled Weight Matrix 2, which results in an output pair in cipherspace. Additionally, as shown in Fig 5(b), a scaling-down operation is required during the decryption conducted in the encryption interface in the application initiator.

Figure 5(c) illustrates the implementation of our element-wise encryption strategy to execute a convolutional neural network (CNN) in cipherspace. Similar to the previous example, the convolution filter is converted to a pair of scaled and encrypted filters having the dimension of $m \times r$. A dot product operation is performed on each encrypted convolution filter of the pair with the scaled input vector. Then a pair of hidden layers in cipherspace is achieved by implementing ReLU operations, sum-pooling, and reshaping. Another dot product operation is executed on each of the hidden layers in the pair with the scaled Weight Matrix 2, which results in an output pair in cipherspace. Since, in this example structure, the encrypted ReLU activation function is realized via the random dropout and the operations of sum-pooling and reshaping are executed in cipherspace using IVHE scheme, no additional scaling is introduced through these operations. Therefore, the final output scaling factor remains lower compared to the previous example fully-connected NN using sigmoid activation function.

b) Matrix-Pair-wise Encryption Strategy: Our matrix-pair-wise encryption strategy is performed on a pair of neighboring weight matrices. To illustrate our strategy, we an example with CNN as shown in Fig. 6. As illustrated in Fig. 6, this CNN has two convolution layers having the dimensions of $m \times 1 \times r$ and $m \times r \times l$, respectively, where r is the number of convolution filters in the first convolutional layer and l is the number of convolution filters in the second convolutional layer.

In our encryption strategy, the dimension of encryption is carefully selected such that there is no dimension mismatch while executing the CNN-based machine learning model in cipherspace. Additionally, the first convolution layer is encrypted via IVHE scheme by leveraging the 3rd dimension, which results in a scaled and encrypted convolutional filter with dimension $m \times 1 \times (r+1)$. Similarly, the second convolution layer is encrypted on the 2nd dimension, which results in a scaled and encrypted convolutional filter with dimension $m \times (r+1) \times l$. By doing so, the convolution operation can be performed as a pair without any dimension mismatch.

C. Learning-Model Fusion Mechanism

As illustrated in Fig. 2, the success of our decentralized and secure computing paradigm requires an efficient learning-model fusion mechanism, with which the application initiator is able to integrate the verified private models and achieve an effective MetaModel for the targeted data-driven task. For designing the learning-model fusion mechanism, it is necessary to treat the computing models, provided by the computing contributors, as separate entities, to ensure the fused structure is dynamic.

Figure 7 illustrates the structure of our proposed fusion mechanism. In our mechanism, the upper-layer feature vectors $\mathbf{f_i}$ from individual verified private models are concatenated to form a concatenated feature layer $\mathbf{f_c}$. As shown in Fig. 7, the upper-layer feature vector $\mathbf{f_i}$ can be the input of the output layer for the private model i. One fully-connected neural network (NN) is designed to fuse the features characterized by the individual private models. This fully-connected NN uses the concatenated feature layer $\mathbf{f_c}$ as its input layer and has a hidden layer $\mathbf{h}$ with the length of $\sum_{i=1}^{n} \mid l_i \mid$, where $\mid l_i \mid$ is the

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/TETC.2020.2983007, IEEE Transactions on Emerging Topics in Computing

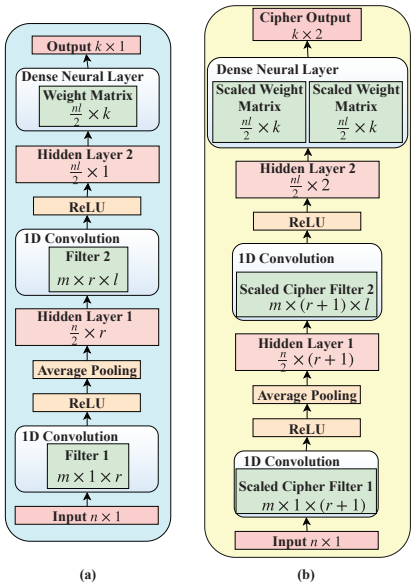


Fig. 6. Illustration of our matrix-pair-wise encryption strategy: (a) a CNN-based machine learning model in plainspace and (b) the corresponding CNN-based machine learning model in ciperspace.

number of the labeled classes in the ith private model. To design the fully-connected NN, it is essential to design its weight matrices $\mathbf{A}$ and $\mathbf{B}$. Currently, we consider two strategies for learning the optimum values for weight matrices $\mathbf{A}$ and $\mathbf{B}$ and designing the learning-model fusion mechanism.

1) Strategy I: In this strategy, the weight matrices $\mathbf{A}$ and $\mathbf{B}$ are initialized randomly without any additional constraints and are optimized via the backpropagation algorithm [3] in which the ith element in the hidden layer $\mathbf{h}$ and the jth element in the output layer $\mathbf{y}$ are calculated as follows:

$$h_j = \sum_{i=1}^{|\mathbf{fc}|} A_{ij}^T \cdot fc_i \tag{8}$$

$$y_j = \frac{exp(\sum_{i=1}^{|\mathbf{h}|} B_{ij}^T \cdot h_i)}{\sum_{k=1}^{d} exp(\sum_{i=1}^{|\mathbf{h}|} B_{ik}^T \cdot h_i)} \tag{9}$$

2) Strategy II: This strategy is developed to achieve two goals: (1) learning the uniqueness of the features characterized by the individual verified private models, and (2) exploring the correlation amongst the features presented by the individual private models. To achieve this goal, a gradual fusion is designed, in which the weight matrix $\mathbf{A}$ is initialized as a concatenated matrix formulated in Eq. (10) and the weight matrix $\mathbf{B}$ is initialized with the concatenation of the identity matrices formulated in Eq. (11).

$$\mathbf{A}_{init} = \begin{bmatrix} \mathbf{W_1} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{W}_2 & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{W}_3 & \dots & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{W}_n \end{bmatrix} \tag{10}$$

where a diagonal weight matrix $\mathbf{W}_i$, which has the dimension of $(\mid \mathbf{f}_i \mid, \mid \mathbf{l}_i \mid)$, is initialized randomly.

$$\mathbf{B}_{init} = \begin{bmatrix} w_{111}=1 & 0 & 0 & \dots & 0 \\ 0 & w_{122}=1 & 0 & \dots & 0 \\ 0 & 0 & w_{133}=1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & w_{1dd}=1 \\ w_{211}=1 & 0 & 0 & \dots & 0 \\ 0 & w_{222}=1 & 0 & \dots & 0 \\ 0 & 0 & w_{233}=1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & w_{2dd}=1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ w_{n11}=1 & 0 & 0 & \dots & 0 \\ 0 & w_{n22}=1 & 0 & \dots & 0 \\ 0 & 0 & w_{n33}=1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & w_{ndd}=1 \end{bmatrix} \tag{11}$$

where n is the number of the verified private models to be fused and d denotes the number of classification labels.

The elements of the weight matrices $\mathbf{A}$ and $\mathbf{B}$ are optimized by using our gradual fusion method that consists of two stages. In the initial stage, only the diagonal non-zeros weights of matrix $\mathbf{A}$ are updated by using backpropagation algorithm, which targets at learning the uniqueness of the features characterized by the individual private models. In the second stage, all of the weights in $\mathbf{A}$ are updated by using backpropagation algorithm, which targets at exploring the correlations between the features characterized by the individual private models. Accordingly, the ith element in the hidden layer $\mathbf{h}$ and the jth element in the output layer $\mathbf{y}$ are calculated as follows:

$$h_j = \sum_{i=1, A_{ij} \in \bar{\mathbf{W}}}^{|\mathbf{fc}|} A_{ij}^T \cdot fc_i + \gamma \sum_{i=1, A_{ij} \notin \bar{\mathbf{W}}}^{|\mathbf{fc}|} A_{ij}^T \cdot fc_i \tag{12}$$

where the parameter γ is set as 0 in the initial stage and as 1 in the final stage, $\bar{\mathbf{W}} = \mathbf{W}_p \leftarrow \sum_{k=1}^{p-1} \mid \mathbf{l}_k \mid < i \leq \sum_{k=1}^{p} \mid \mathbf{l}_k \mid$.

$$y_j = \frac{exp(\sum_{i=1, B_{ij}=w_{jpp}}^{|\mathbf{h}|} B_{ij}^T \cdot h_i + \gamma \sum_{i=1, B_{ij} \neq w_{jpp}}^{|\mathbf{h}|} B_{ij}^T \cdot h_i))}{\sum_{k=1}^{d} exp(\sum_{i=1, B_{ik}=w_{ipp}}^{|\mathbf{h}|} B_{ik}^T \cdot h_i + \gamma \sum_{i=1, B_{ik} \neq w_{ipp}}^{|\mathbf{h}|} B_{ik}^T \cdot h_i)} \tag{13}$$

where $p \in \{1, 2, 3 \dots, d\}$, and d is the number of labeled classes.

D. Blockchain-based Middleware

Ethereum Blockchain-based middleware is designed to automatically control, manage and secure the system processes. First of all, Proof-of-Work (PoW) consensus protocol of blockchain provides an unsupervised secure environment where the authoritative agents are no longer necessary for the distributed computing. Additionally, the distributed database of blockchain provides an immutable ledge of any events happened in time order on the system. It is convenient to trace the inputs and outputs of each event on the blockchain ledge. Furthermore, blockchain smart contract enables the automation of system processes including the training, verification, transaction, and fusion processes of the decentralized and cooperative learning.

VI. Security Analysis

A. Analysis of Blockchain-Powered Integrity

Our proposed decentralized and secure computing paradigm achieves satisfactory integrity by exploiting blockchain technology with Proof-of-Work (PoW) consensus protocol. To analyze the integrity security achieved by our proposed paradigm, we specify the following threat model for the attackers:

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/TETC.2020.2983007, IEEE Transactions on Emerging Topics in Computing

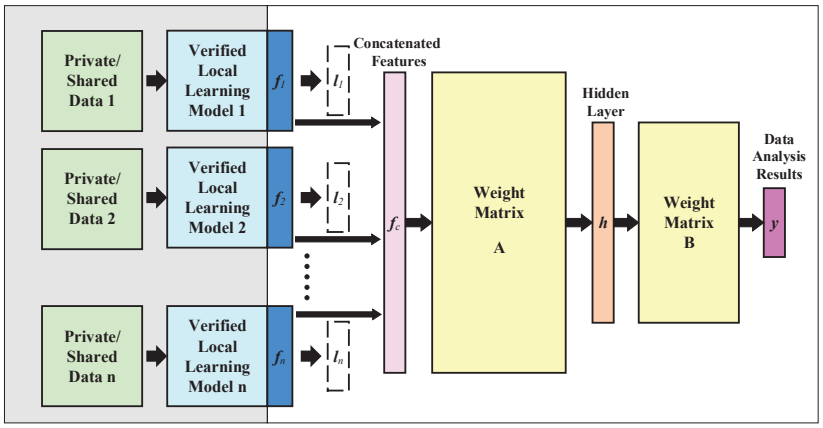


Fig. 7. Illustration of our proposed learning-model fusion mechanism.

1) Let n be total mining power of all the anonymous nodes, each of which has equal mining power, in the blockchain platform of our paradigm. The attackers compromise the nodes with f mining power via Byzantine fault.
2) The attackers aim to maximize the possibility that system can not achieve consensus by convincing other nodes in the blockchain platform to believe the faked chain data at or after the block confirmation time.
3) The attackers cannot compromise the communication channels for the blockchain platform. In other words, the attackers are not able to launch an Eclipse attack [44].

Based on the threat model, we are able to get that the probability that the compromised nodes successfully discover the next block in a block interval can be calculated as: $q = \frac{f}{n}$ and the success rate that the honest nodes discover the next block in a block interval is $p = 1 - q$. The competition between the compromised and honest nodes can be formulated as a special case of Gambler's Ruin problem [45]. By doing so, we are able to calculate the probability that the compromised nodes can catch up z blocks as follows:

$$q_z = \begin{cases} 1 & if\ p \leq q \\ (\frac{q}{p})^z & if\ p > q \end{cases} \tag{14}$$

Considering that PoW consensus protocol only allows the longest chain to be accepted as a trusted one, where the latest block must be verified successfully at each node that receives it, the compromised nodes have to catch up at least z blocks to disrupt the blockchain successfully at anytime after the z confirmation block intervals. Assuming the block confirmation time is T, the time used by the honest nodes to generate z blocks can be calculated as $z \cdot \frac{T}{p} = \frac{zT}{p}$. During this time interval, the expected number of blocks generated by the compromised nodes are $\frac{zT}{p} \cdot \frac{q}{T} = \frac{zq}{p}$. Based on the statements in [13], it is reasonable to assume that the valid newly generated blocks in a certain time interval is independent and the success rate of discovering new blocks is positively correlated to the duration of the time interval even though the discovery of new blocks is a transient state. Under these assumptions, we are able to model the probability that the compromised nodes discover k blocks within a certain time interval $\frac{zT}{p}$ as a Poisson distribution. Therefore, the probability that the compromised nodes can generate k blocks within time interval $\frac{zT}{p}$ can be calculated as:

$$P(k, \lambda) = \frac{\lambda^k e^{-\lambda}}{k!}. \tag{15}$$

Based on Eqs. (14) and (15), we can obtain the total probability that the compromised nodes can generate z blocks after $\frac{z}{p}$ confirmation time intervals as

$$\begin{aligned} \mathcal{P} &= P(0, \lambda)q_z + P(1, \lambda)q_{z-1} + \cdots + P(z, \lambda)q_0 \\ &= \sum_{k=0}^{\infty} \frac{\lambda^k e^{-\lambda}}{k!} \begin{cases} (\frac{f}{n-f})^{z-k} & if\ k \leq z \\ 1 & if\ k > z \end{cases} \end{aligned} \tag{16}$$

where $\lambda = \frac{zq}{p}$. From Eq. (16), it is clear that we are able to ensure a satisfactory integrity by determining the critical block confirmation time interval as long as we are able to estimate the upper bound of mining power f of the nodes compromised by the attacker. The upper bound of f can be estimated by effectively monitoring the behavior of the blockchain platform and efficiently enhance the security of certain number of the blockchain nodes with available resources.

1) Numerical Demonstration: We demonstrate the integrity analysis stated above via a numerical demonstration of the blockchain PoW consensus protocol using Python. We assume there are 1000 active nodes in the blockchain platform. Additionally, we assume that the attacker is capable of randomly compromising m of the 1000 nodes in the blockchain platform, where $m = 50, 100, 200, 300, 340$, or 400. Therefore, the probability that the compromised nodes successfully discover the next block in a block interval is $q = \frac{m}{1000} = 5\%$, 10%, 20%, 30%, 34%, or 40%. The theoretical result, based on Eqn. (16), is shown in Fig. 8(a). To achieve the theoretical result, the attacker is assumed to have the resources to launch the attacks for infinite number of block intervals. Considering the attackers have limited resources in practical situation, in Fig. 8(b), we assume the attacker can only afford to launch the attacks for 1000 block intervals and obtain the result via Monte Carlo simulation. Comparing Figs. 8(a) and (b), we can observe that the practical simulation result has the same tendency as the theoretical result. Additionally, the attack success rate in the practical result is less than that in the theoretical result, which is reasonable because of the limited resources of the attacker. From Figs. 8(a) and (b), we can also obtain that, when the attacker controls 34% mining power, 29 block intervals are required theoretically and 19 block intervals are required in the considered practical situation to achieve confidence above 99%. This result also demonstrates the advantage of the blockchain-powered security solution exploited in our computing paradigm compared with Practical Byzantine Fault Tolerance (PBFT)-based solution [46], in which the security cannot be maintained if more than $\frac{1}{3}$ nodes are compromised.

B. Analysis of The Robustness to Lazy or Dishonest Contributors

One challenge of cloud computing is that cloud servers can be lazy (or dishonest) and use less amount of computing power 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/TETC.2020.2983007, IEEE Transactions on Emerging Topics in Computing

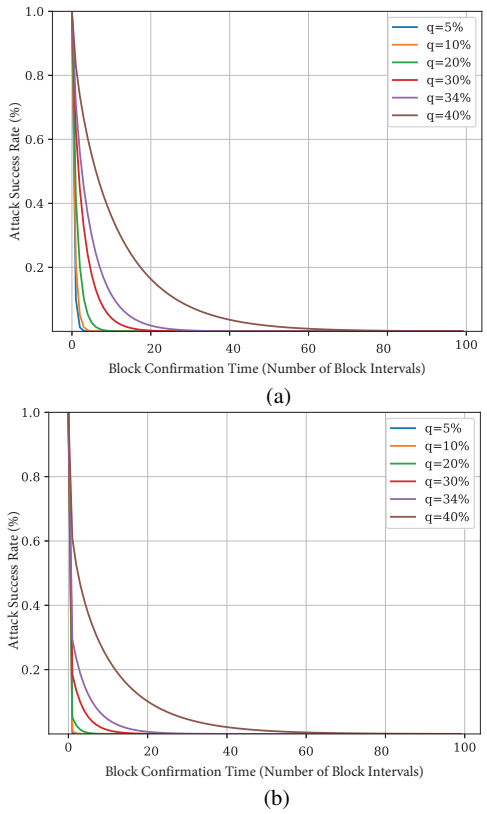


Fig. 8. Attack Success Rate with different compromised mining power: (a) practical result and (b) theoretical result.

provide inaccurate results. The motivations of these lazy or dishonest cloud computing include (1) the financial savings achieved by the reduced computing power and (2) the malicious purposes. Verifiable computing is an active research area to address this issue [47]. The computing paradigm developed in our work can provide a solution to address this issue from another prospective. In our proposed paradigm, if the application initiators are considered as the clients, the associated computing and verification contributors can be considered all together as the cloud. The threat model of the lazy or dishonest cloud in the context of our work is specified as follows:

1) The computing contributors claim their private models, which are not well trained, or
2) Let $\mathcal{N}_v$ be the number of passively selected verification contributors. There are up to $\lfloor \frac{1}{2}\mathcal{N}_v \rfloor$ verification contributors determine the quantitative contributions of the claimed private models in an inappropriate manner, and
3) Either the computing contributors or verification contributors perform appropriately.

If the selected verification contributors perform appropriately, they will evaluate the quantitative contributions of the private models, which are claimed by the computing contributors, according to the workflow shown in Fig. 3(c). As shown in Fig. 3(c), the verification contributors first download the private models claimed by the computing contributors from the IPFS decentralized storage and then start to verify the contributions of the downloaded private models sequentially. The verification procedure consists of three main steps: (1) merging the downloaded new model with the MetaModel by implementing the model fusion strategies described in Section V-C and then achieving a temporarily updated MetaModel, (2) comparing the performance of the MetaModel and that of its temporarily updated version, and (3) making decision on whether the new model has satisfactory contribution. If the performance of the temporarily updated MetaModel, such as inference accuracy, is higher, the verification contributors conclude that the contribution of this new model is satisfactory. Otherwise, the verification contributors conclude that the new model is not useful. Furthermore, our computing paradigm does not allow the transactions of the new models that do not have satisfactory contributions. If there exist new models claimed by lazy or dishonest computing contributors, since these new models are not well trained locally, they cannot perform effectively during the verification procedure. It is reasonable to expect the majority of the associated verification contributors conclude that these new models do not have satisfactory contributions and there are no transactions to be established between the application initiator and the lazy or dishonest computing contributors. In other words, if the verification contributors perform appropriately, based on our threat model, our computing paradigm is able to prevent the interruption of the lazy or dishonest computing contributors. If there exist lazy or dishonest verification contributors, our majority voting mechanism for verification procedure is able to evaluate the contributions of the claimed private models appropriately as long as the majority of the verification contributors still perform in an appropriate manner. As stated in our threat model, the number of the lazy or dishonest verification contributors is up to $\lfloor \frac{1}{2}\mathcal{N}_v \rfloor$, where $\mathcal{N}_v$ is the number of passively selected verification contributors for the given task. Therefore, it is reasonable to claim our computing paradigm is robust to the existence of the lazy or dishonest verification contributors.

C. Analysis of The Robustness to Honest-but-Curious Contributors

Another challenge of cloud computing is that cloud server can be honest-but-curious. This kind of cloud server participates in the computation legitimately while attempting to discover the private information from the received messages [42], [48]. Our computing paradigm can provide a solution to mitigate this issue from another prospective. In our proposed paradigm, private information is mainly from the local data possessed by computing contributors. Considering the fact that the locally trained machine learning models, rather than the raw data, of the computing contributors are shared in our computing paradigm, the behavior of the honest-but-curious verification contributors and/or application initiators are obstructed in a certain level. Although the honest-but-curious contributors may launch inference attacks to infer from the shared machine learning models about the private raw data [49], the homomorphic encryption interface designed in our computing paradigm hinders the access of the verification contributors to the shared model in plainspace and prevents the honest-but-curious verification contributors from launching inference attacks. Furthermore, during the transaction of the private models between the associated computing contributors and the application initiators, the application initiators obtain the access to the shared model in plainspace by providing the computing contributors with financial compensations. Therefore, the private data owned by the computing contributors may be vulnerable to the inference attacks launched by the honest-but-curious contributors. However, since our blockchain-powered computing paradigm enables the contributors to remain anonymous while participating in the data-driven task, the disclosure of the relation between the data generated from the attack and the identify of the associated computing contributors is prevented effectively.

D. Analysis of The Effectiveness of Our Homomorphic Encryption Interface

The homomorphic encryption interface is one of the essential components in computing layer to ensure the security of computing in our paradigm. In this section, we analyze the effectiveness of

our homomorphic encryption interface by analyzing the accuracy of artificial neural network (ANN) inference in the cipherspace with homomorphic encrypion.

First, we consider that one of the neural network layers in the ANN model is encrypted. We use two main types of ANN models, dense neural network (DNN) and convolutional neural network (CNN), for the analysis. The mechanism of homomorphic encryption can be formulated by the following encryption equation:

$$\mathbf{Sc}^x = w\mathbf{x} + \mathbf{e} \tag{17}$$

where $\mathbf{S}$ is the private-key matrix, $\mathbf{x}$ is the plaintext vector, $\mathbf{c}^x$ is the ciphertext vector corresponding to plaintext vector $\mathbf{x}$, $\mathbf{e}$ is a randomization integer term introduced to enable the encryption, which has elements with values smaller than w, and w is a large-value integer that controls the appropriate ratio between the plaintext and the introduced randomization. Decrypted plaintext vector $\bar{\mathbf{y}}$ is obtained from a cipher vector $\mathbf{c}^y$ according to the following decryption equation:

$$\bar{\mathbf{y}} = \lceil \frac{\mathbf{Sc}^y}{w} \rfloor \tag{18}$$

If the ANN model is a DNN that comprises fully connected layers of artificial neurons (AN). In a DNN structure, each AN mainly performs two operations, weighted sum and activation function. Let $\mathbf{y}$ denote the vector of weighted sums of a layer of ANs. $\mathbf{y}$ can be calculated via a matrix-vector multiplication as follows:

$$\begin{aligned} \left[\mathbf{y}\right]_{m\times 1} &= \left[\mathbf{x}_1, \quad \mathbf{x}_2, \quad \cdots, \quad \mathbf{x}_n\right]_{m\times n} \left[\mathbf{g}\right]_{n\times 1} \\ &= \left[g_1\mathbf{x}_1 + g_2\mathbf{x}_2 + \cdots + g_n\mathbf{x}_n\right]_{m\times 1} \end{aligned} \tag{19}$$

where $\mathbf{g}$ is the output of the previous layer of ANs and $\mathbf{x}_k$ is the weight vector that connects the output of the previous layer to the kth AN of the current layer. From Eq. (19), we can obtain that the weighted sum of the kth AN can be calculated as:

$$y_k = g_1x_{k1} + g_2x_{k2} + \cdots + g_nx_{kn} \tag{20}$$

The representation of y_k in cipherspace with homomorphic encryption can be calculated as follows:

$$\left[\mathbf{c}^y\right]_{(m+1)\times 1} = \left[\mathbf{c}_1^x, \quad \mathbf{c}_2^x, \quad \cdots, \quad \mathbf{c}_n^x\right]_{(m+1)\times n} \left[\mathbf{g}\right]_{n\times 1} \tag{21}$$

where $\mathbf{g}$ remains the output vector of the previous layer in plainspace, $\mathbf{c}_k^x$ is the encrypted cipher-domain representation of the vector $\mathbf{x}_k$, and $\mathbf{c}^y$ is the encrypted cipher-domain representation of the vector $\mathbf{y}$. Additionally, $\mathbf{c}_i^x$ and $\mathbf{x}_i$ satisfies the relations defined in Eq. (17).

By multiplying both sides of Eq. (21) with $\mathbf{S}$, and substituting the result to Eq. (17), we can obtain the following equation:

$$\begin{aligned} \left[\mathbf{Sc}^y\right]_{m\times 1} &= \left[\mathbf{Sc}_1^x, \quad \mathbf{Sc}_2^x, \quad \cdots, \quad \mathbf{Sc}_n^x\right]_{m\times n} \left[\mathbf{g}\right]_{n\times 1} \\ &= \left[w\mathbf{x}_1 + \mathbf{e}_1, \quad w\mathbf{x}_2 + \mathbf{e}_2, \quad \cdots, \quad w\mathbf{x}_n + \mathbf{e}_n\right]_{m\times n} \\ &\quad \left[\mathbf{g}\right]_{n\times 1} \\ &= \left[g_1w\mathbf{x}_1 + g_2w\mathbf{x}_2 + \cdots + g_nw\mathbf{x}_n\right]_{m\times 1} + \\ &\quad \left[g_1\mathbf{e}_1 + g_2\mathbf{e}_2 + \cdots + g_n\mathbf{e}_n\right]_{m\times 1} \\ &= w\left[g_1\mathbf{x}_1 + g_2\mathbf{x}_2 + \cdots + g_n\mathbf{x}_n\right]_{m\times 1} + \\ &\quad \left[g_1\mathbf{e}_1 + g_2\mathbf{e}_2 + \cdots + g_n\mathbf{e}_n\right]_{m\times 1} \end{aligned} \tag{22}$$

Based on Eqs. (18) and (22), we can obtain the decrypted weighted sum $\bar{\mathbf{y}}$ as follows:

$$\begin{aligned} \left[\bar{\mathbf{y}}\right]_{m\times 1} = \Bigg\lceil & \left[g_1\mathbf{x}_1 + g_2\mathbf{x}_2 + \cdots + g_n\mathbf{x}_n\right]_{m\times 1} + \\ & \frac{1}{w}\left[g_1\mathbf{e}_1 + g_2\mathbf{e}_2 + \cdots + g_n\mathbf{e}_n\right]_{m\times 1} \Bigg\rfloor \end{aligned} \tag{23}$$

From Eq. (23), we can obtain the equation for the decypted weighted sum of the kth AN as follows:

$$\begin{aligned} \bar{y}_k = \lceil & (g_1x_{i1} + g_2x_{i2} + \cdots + g_nx_{in}) + \\ & \frac{1}{w}(g_1e_{k1} + g_2e_{k2} + \cdots + g_ne_{kn}) \rfloor \end{aligned} \tag{24}$$

To guarantee the high accuracy of DNN inference in cipherspace with homomorphic encryption, we have $\frac{1}{w}|g_1e_{k1}+g_2e_{k2}+\cdots+g_ne_{kn}| < 0.5$ for all k. Letting g_B and e_B to denote the upper bounds of $|g_i|$ and $|e_{ki}|$, respectively, we can achieve the following inequality:

$$\frac{1}{w}|g_1e_{k1} + g_2e_{k2} + \cdots + g_ne_{kn}| < n \cdot g_B\frac{e_B}{w} < 0.5 \tag{25}$$

Therefore, to ensure the accuracy of DNN inference in cipherspace, our homomorphic encryption interface is designed to satisfy the following upper bound for the ratio, $\frac{e_B}{w}$, for the weighted sum operation:

$$\frac{e_B}{w} < \frac{0.5}{n \cdot g_B} \tag{26}$$

Considering that the random error introduced for encryption needs to be an appropriately small integer, both $\mathbf{A}$ and $\mathbf{E}$ terms used to calculate the public key matrix $\mathbf{M}$ as shown in Eq. (3) have small values. Therefore, the encryption achieved by multiplying a plaintext of length n with $\mathbf{M}$ is less likely to change the sign of a leading n terms of ciphertext. Considering this property, we can perform rectifier operation in cipherspace to achieve the encrypted result of the corresponding rectifier activation function in plainspace, which has a low probability of error effecting the plaintext values that are close to 0. In other words, this error can be considered as negligible for the overall accuracy of a DNN structure.

If the ANN model is a CNN, the additional operations needed to be implemented include (a) convolution operation and (a) pooling operation.

a) Convolution Operation: To simplify the analysis, we consider that the CNN has 1D convolution layers. The authors would like to mention that the related analysis can be generalized to 2D convolutions. The input vector to the convolution layer is denoted as $\mathbf{g}$, the pth convolution kernel is denoted as $\mathbf{K}^p$, and the feature map obtained through the pth kernel is denoted by $\mathbf{y}^p$. Letting the size of each kernel to be $2k$ and the number of the kernels to be P, the ith values of the pth feature map can be calculated via implementing the convolution between the input and the pth kernel as follows:

$$y_{pi} = \sum_{j=-k}^{k} g_{i+j}K_{p(j+k)} \tag{27}$$

Similarly, the ith element of all the P feature vectors can be calculated as follows:

$$\begin{bmatrix} y_{1i} \\ y_{2i} \\ \vdots \\ y_{Pi} \end{bmatrix}_{P\times 1} = \sum_{j=-k}^{k} g_{i+j} \begin{bmatrix} K_{1(j+k)} \\ K_{2(j+k)} \\ \vdots \\ K_{P(j+k)} \end{bmatrix}_{P\times 1} \tag{28}$$

Letting $\mathbf{c}_j^K$ and $\mathbf{c}_i^y$ to be the encrypted representations of $[K_{j1}, K_{j2}, \cdots, K_{jP}]^T$ and $[y_{i1}, y_{i2}, \cdots, y_{iP}]^T$, respectively, the convolution operation in cipherspace can be formulated as:

$$\left[\mathbf{c}_i^y\right]_{(p+1)\times 1} = \sum_{j=-k}^{k} g_{i+j}\left[\mathbf{c}_{(j+k)}^K\right]_{(p+1)\times 1} \tag{29}$$

By multiplying both side of Eq. (29) with $\mathbf{S}$, we can obtain:

$$\left[\mathbf{Sc}^y\right]_{P\times 1} = \sum_{j=-k}^{k} g_{i+j}\left[\mathbf{Sc}_{(j+k)}^K\right]_{P\times 1} \tag{30}$$

Using Eq. (17), Eq. (30) can be represented as:

$$\left[\mathbf{Sc}^{y}\right]_{P\times 1}=\sum_{j=-k}^{k} g_{i+j}\begin{bmatrix} wK_{1(j+k)}+e_{1(j)} \\ wK_{2(j+k)}+e_{2(j)} \\ \vdots \\ wK_{P(j+k)}+e_{Pj}\end{bmatrix}_{P\times 1} \tag{31}$$

Therefore, according to the Eq. (18), the ith decrypted value of the pth feature map can be calculated as follows:

$$\begin{aligned}\bar{y}_{pi}&=\left\lceil \sum_{j=-k}^{k} g_{i+j}[K_{p(j+k)}+\frac{e_{pj}}{w}]\right\rfloor \\ &=\left\lceil \sum_{j=-k}^{k} x_{i+j}K_{p(j+k)}+\frac{1}{w}\sum_{j=-k}^{k} g_{i+j}e_{pj}\right\rfloor\end{aligned} \tag{32}$$

To guarantee the accuracy of convolution operation in cipherspace, we need to ensure $\frac{1}{w}|\sum_{j=-k}^{k} g_{i+j}e_{pj}|<0.5$ for all p and j.

Letting g_B and e_B be the upper bound of $|g_i|$ and $|e_{pi}|$, respectively, we can obtain the followings:

$$\frac{1}{w}|\sum_{j=-k}^{k} g_{i+j}e_{pj}|<2kg_B\frac{e_B}{w}<0.5 \tag{33}$$

Therefore, to ensure the accuracy of CNN inference in cipherspace, our homomorphic encryption interface is designed to satisfy the following upper bound for the ratio, $\frac{e_B}{w}$, for convolution operation:

$$\frac{e_B}{w}<\frac{0.5}{2kg_B} \tag{34}$$

b) Pooling Operation: In our work, the pooling operation of CNN models is considered to be sum pooling that reduces the size of the feature map by half. Letting y_{pi} and z_{pi} denote the ith element of the pth input and output feature maps of the pooling operation, respectively, we can calculate z_{pi} as follows;

$$z_{pi}=y_{p(2i)}+y_{p(2i+1)} \tag{35}$$

This can be represented as a vector operation on all P feature maps as follows;

$$\begin{bmatrix} z_{1i} \\ z_{2i} \\ \vdots \\ z_{Pi}\end{bmatrix}_{P\times 1}=\begin{bmatrix} y_{1(2i)} \\ y_{2(2i)} \\ \vdots \\ y_{P(2i)}\end{bmatrix}_{P\times 1}+\begin{bmatrix} y_{1(2i+1)} \\ y_{2(2i+1)} \\ \vdots \\ y_{P(2i+1)}\end{bmatrix}_{P\times 1} \tag{36}$$

Therefore, the pooling operation in cipherspace can be formulated as:

$$\left[\mathbf{c}^{z}{}_{i}\right]_{(P+1)\times 1}=\left[\mathbf{c}^{y}{}_{2i}\right]_{(P+1)\times 1}+\left[\mathbf{c}^{y}{}_{2i+1}\right]_{(P+1)\times 1} \tag{37}$$

By multiplying the both sides of Eq. (37) with $\mathbf{S}$ and substituting the result to Eq. (17), we can get:

$$\begin{aligned}\left[\mathbf{Sc}_{i}^{z}\right]_{P\times 1}&=\left[\mathbf{Sc}_{2i}^{y}\right]_{P\times 1}+\left[\mathbf{Sc}_{2i+1}^{y}\right]_{P\times 1} \\ &=\left[w\mathbf{y}_{2i}+\mathbf{e}_{2i}\right]_{P\times 1}+\left[w\mathbf{y}_{2i+1}+\mathbf{e}_{2i+1}\right]_{P\times 1} \\ &=w\left[\mathbf{y}_{2i}+\mathbf{y}_{2i+1}\right]_{P\times 1}+\left[\mathbf{e}_{2i}+\mathbf{e}_{2i+1}\right]_{P\times 1}\end{aligned} \tag{38}$$

Based on Eq. (18), the ith pooling output of the pth feature map can be calculated as:

$$\bar{z}_{pi}=\lceil y_{p(2i)}+y_{p(2i+1)}+\frac{1}{w}(e_{2i}+e_{2i+1})\rfloor \tag{39}$$

To ensure the accuracy of the pooling operation in cipherspace, we have

$$\frac{1}{w}|e_{2i}+e_{2i+1}|<0.5 \tag{40}$$

for all i. Letting e_B to be the upper bound for $|e_i|$, we can obtain the following conclusion:

$$\frac{1}{w}|e_{2i}+e_{2i+1}|<\frac{2e_B}{w}<0.5 \tag{41}$$

Therefore, to ensure the accuracy of CNN inference in cipherspace, our homomorphic encryption interface is designed to satisfy the following upper bound for the ratio $\frac{e_B}{w}$, for pooling operation:

$$\frac{e_B}{w}<0.25 \tag{42}$$

Next, we continue to analyze the accuracy of the ANN inference in the cipherspace with homomorphic encryption when the ANN structure have multiple encypted layers. From the analysis above, it is clear that the layers in both DNN and CNN can be formulated based on a vector multiplication operation. If there is only one encrypted layer in the ANN model, one vector in this vector multiplication is encrypted and the other one is a plaintext. When there are more than one encrypted layer in the ANN model, there exist at least one vector multiplication, in which both vectors are in cipherspace. This kind of vector multiplication can be represented in plainspace as follows:

$$\mathbf{y}=\mathbf{x}^{T}\mathbf{g} \tag{43}$$

where $\mathbf{y}$, $\mathbf{x}$, and $\mathbf{g}$ are column vectors. The corresponding operation in ciperspace can be formulated as:

$$\mathbf{c}^{y}=\lceil\frac{vec[\mathbf{c}^{x}(\mathbf{c}^{g})^{T}]}{w}\rfloor \tag{44}$$

where $\mathbf{c}^{y}$, $\mathbf{c}^{x}$, and $\mathbf{c}^{g}$ are the cipher domain representation for vectors, $\mathbf{y}$, $\mathbf{x}$, and $\mathbf{y}$, respectively, and $vec(\cdot)$ denotes the vectorization operation, where the columns of a matrix are concatenated to result in a vector. The final private key for decrypting $\mathbf{c}^{y}$ can be calculated as $\mathbf{S}'=[vec(\mathbf{S}^{T}\mathbf{S})]^{T}$. For simplicity of analysis, we assume that the result of $\frac{vec[\mathbf{c}^{x}(\mathbf{c}^{g})^{T}]}{w}$ is an integer vector. Based on Eq. (17), we can obtain the following result:

$$\mathbf{S}'\frac{vec[\mathbf{c}^{x}(\mathbf{c}^{g})^{T}]}{w}=w\mathbf{y}+\mathbf{e}_{y} \tag{45}$$

where $\mathbf{e}_{y}$ is the overall error. Equation (45) can be represented as follows:

$$\begin{aligned}vec(\mathbf{S}^{T}\mathbf{S})^{T}\frac{vec[\mathbf{c}^{x}(\mathbf{c}^{g})^{T}]}{w}&=\frac{1}{w}[(\mathbf{c}^{x})^{T}\mathbf{S}^{T}]\mathbf{Sc}^{g} \\ &=\frac{1}{w}(w\mathbf{x}+\mathbf{e}_{x})^{T}(w\mathbf{g}+\mathbf{e}_{g}) \\ &=w\mathbf{x}^{T}\mathbf{g}+\mathbf{x}^{T}\mathbf{e}_{g}+\mathbf{e}_{x}^{T}\mathbf{g}+\frac{\mathbf{e}_{x}^{T}\mathbf{e}_{g}}{w}\end{aligned} \tag{46}$$

where $\mathbf{e}_{x}$ and $\mathbf{e}_{g}$ are random errors introduced for encrypting x and g, respectively. Based on Eqs. (43) and (45), we can formulate the overall error $\mathbf{e}_{y}$ in the following:

$$\mathbf{e}_{y}=\mathbf{x}^{T}\mathbf{e}_{g}+\mathbf{e}_{x}^{T}\mathbf{g}+\frac{\mathbf{e}_{x}^{T}\mathbf{e}_{g}}{w} \tag{47}$$

From Eq. (47), it is clear that the overall error is upper bounded if $\mathbf{e}_{x}$ and $\mathbf{e}_{g}$ are upper bounded. In the analysis for the scenario, where ANNs have only one encrypted layer, we have shown that $\mathbf{e}_{x}$ and $\mathbf{e}_{g}$ are upper bounded in the appropriate design.

VII. Simulation Results

In this section, we evaluate the performance of our proposed blockchain-powered decentralized and secure computing paradigm by considering five case studies. To obtain the computation performance of our platform, we develop a Blockchain-powered Software Defined Network (SDN)-based testbed as detailed in Section VII-A.

A. Blockchain-Powered SDN-Based Testbed

Figure 9 (a) shows a picture of our Blockchain-powered SDN-based testbed, in which the computing nodes, including one application initiators, three computing contributors, and three verification contributors, are simulated by using two popular embedded systems, Raspberry Pi and NVIDIA Jetson. The communications and cooperative learning amongst the computing nodes are supported by the blockchain-based middleware and the SDN-based peer-to-peer networking layer. The integration of the computing layer, blockchain-based middleware, and the SDN-enabled networking layer in our computing testbed are illustrated in Fig. 10. Each computing node, which is simulated via Raspberry Pi or NVIDIA Jetson, has (1) one or multiple blockchain clients to interact with the blockchain-based middleware, (2) multiple ethernet ports to interact with the SDN-enabled networking layer, and (3) one or more decentralized storage (DS) clients to enable the decentralized storage systems.

In our testbed, the blockchain-based middleware is developed by exploiting a decentralized application (DApp) on the Ethereum blockchain platform, in which the engine and language of smart contract are Ethereum Virtual Machine (EVM) and Solidity, respectively. The consensus protocol is set to be Proof-of-Work (PoW). In our simulations, the default mining difficulty is changed to 0x80000 to 0x1000. Meanwhile, we also change the difficulty increment function $CalcDifficulty()$ of the Ethereum client $geth$ to a constant integer 1 such that mining difficulty increases 1 after one block generation. By doing so, it is reasonable to consider that the block interval is constant throughout the simulations. Additionally, in our simulations, the block interval is tested as around 10s. All the nodes in blockchain platform are pre-funded with a small number of tokens so that they can pay the gas fee to deploy and call the smart contract for decentralized and secure computing. Furthermore, the smart contract in Ethereum is written by Solidity and the blockchain event listener client is leveraged to provide the interface of the logging facilities of the Ethereum virtual machine. Every computing node, which is willing to participate in a given data-driven task via the blockchain smart contract, is required to send a transaction through Ethereum blockchain to contract with a considerate amount of security deposit of tokens. By doing so, the smart contract is able to recognize all the participants and forward the tasks such as training and verification to the participants appropriately. The deployment of smart contract is considered as a transaction as well, which provides the other nodes with the address necessary to access the smart contract. As shown in Fig 10, the DS system in our testbed is realized by using IPFS that utilizes the peer-to-peer network to share and store hypermedia on the Internet. Since the whole simulation is implemented on a private network, a shared swarm key is used to authorize reliable nodes to join IPFS. Furthermore, our computing layer is developed in Python. Web3.py and ipfs-api.py provide access to Ethereum blockchain and IPFS clients to realize the transactions and file sharing, respectively. Tensorflow is adopted for the task of machine learning in our testbed.

To further demonstrate our testbed, the screenshot of the terminals for a computing initiator, a computing contributor, and a verification contributor during one experiment are shown in Figs. 9(b)-(d), respectively. As shown in Fig. 9(b), the application initiator's operation comprises three main processes that are identified with three red rectangular boxes: (1) being initialized to publish one data-driven task, characterize the objectives and constraints of the task via a smart contract, upload the data to be shared to IPFS, and share the public key generated via homomorphic encryption interface, (2) receiving the verified models that are concluded according to the majority voting of all the selected verification contributors, and (3) implementing decryption and fusion on the received verified models. As shown in Fig. 9(c), the computing contributor's operation mainly consists of four processes: (1) participating the data-driven task, (2) terminating the training task when the accuracy meets the given/self-defined criterion, (3) encrypting the achieved private model via the homomorphic encryption interface and uploading the encrypted private model to IPFS, and (4) announcing the encrypted private model for assessment. As shown in Fig. 9(d), the verification contributor's operation includes three main processes: (1) being passively and randomly selected and initialized for the given data-driven task, (2) receiving all the announced private models, and (3) verifying the received private models according to the criterion defined in smart contract and publishing the verification results.

B. Case Study I

In this case study, we focus on evaluating the performance of our proposed blockchain-powered decentralized and secure computing paradigm in a secure environment. In other words, the functionality of the homomorphic encryption interface is not considered. We assume one application initiator publishes a data-driven task on classifying the 10 image classes provided by the MNIST handwritten digit database [50] and three computing contributors, and three verification contributors, which are considered to be passively and randomly selected, participate in the task. We consider that each computing contributor develops its own CNN-based machine learning model whose structure parameters are summarized in Table II. As shown in Table III, the local data available to each computing contributor only present a partial view of the whole dataset. The accuracy of the CNN models developed by the individual computing contributors for classifying their local verification data is shown in Fig. 11, all of which are above 90 %. In this scenario, the criterion of determining whether the local learning model is trained successfully is achieving the accuracy of 90 %. Therefore, all of the three private CNN models are encrypted and published to the three verification contributors for quantitative verification. After receiving the verified private models according to the majority voting amongst the verification contributors, the application initiator fuses the models by using the two strategies described in Section V-C to achieve the MetaModel. Assuming that the application initiator decrypts and fuses the private models as soon as receiving them, the classification accuracy achieved by the MetaModel is shown in Fig. 12. From Fig. 12, it is clear that the classification accuracy increases as more verified private models are fused. This is reasonable since the individual private models are achieved using the local data that only characterize the partial features of the whole dataset. Furthermore, it can also be seen from Fig. 12 that Fusion Strategy II slightly outperforms the Fusion Strategy I when fusing multiple verified private models.

C. Case Study II

In this case study, we evaluate the performance of the proposed cooperative machine learning platform when there exist lazy or dishonest computing contributors with the threat model specified in Section VI-B. We consider a data-driven task that is similar to the one considered in Case Study I. We assume that all of the three selected verification contributors adopt Strategy II for model fusion during quantitative verification and the computing contributors possess the local datasets as described in Table III and the private machine learning models with the structures shown in Tabel II. We also assume that a honest verification contributor concludes a positive vote if the verification accuracy is above 70% for the first received private model or if the accuracy increases after fusing the received private models. In the simulation, we consider that Computing Contributors 1 and 3

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/TETC.2020.2983007, IEEE Transactions on Emerging Topics in Computing

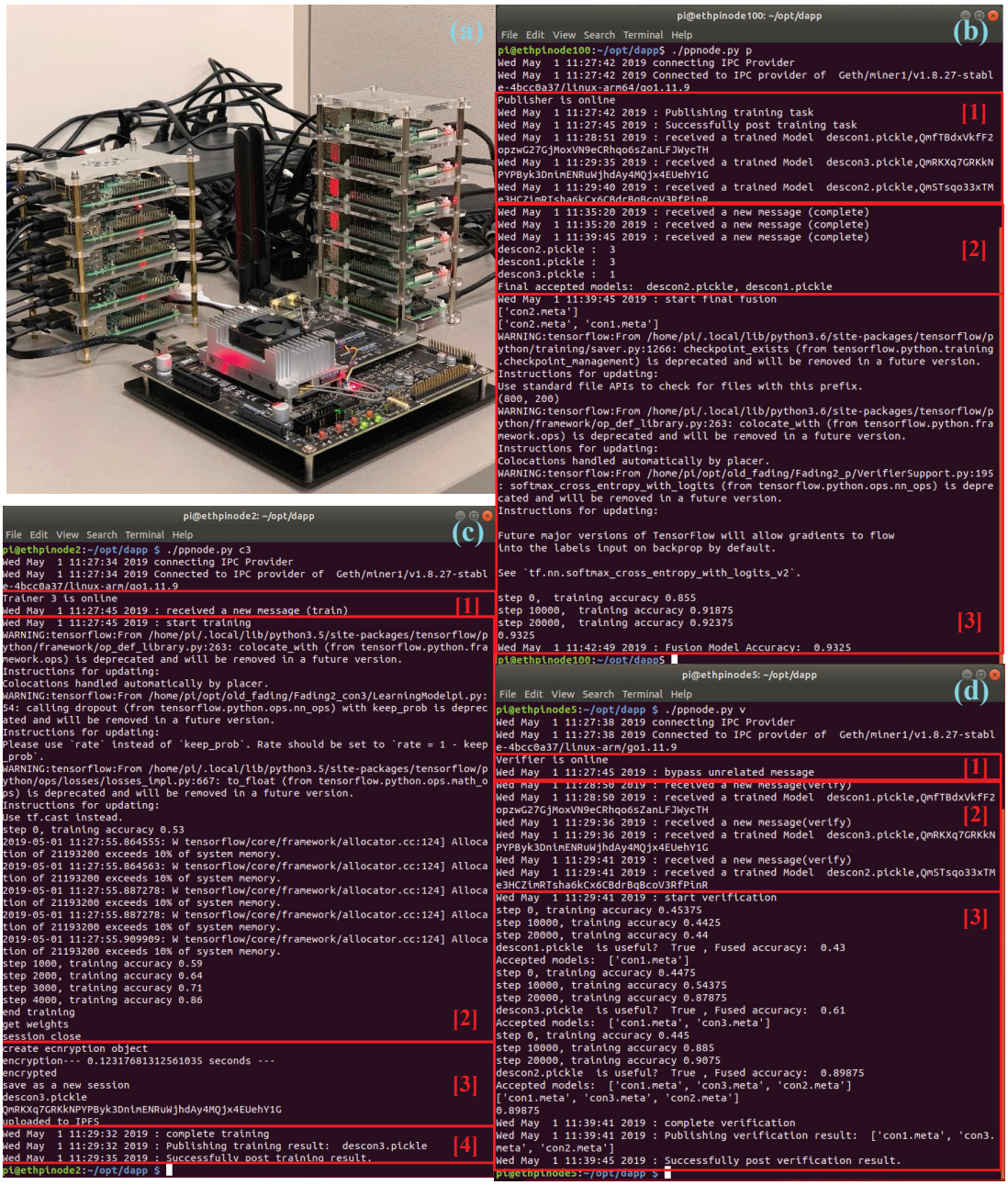


Fig. 9. (a) A picture of the testbed, and the terminals of (b) Application Initiator, (c) Computing Contributor, and (d) Verification Contributor.

do not perform appropriately and claim their private models that are not well trained. We run the simulation for 10 independent trials and achieve the results detailed in Table IV, where i indicates that Private Model i is claimed by a honest computing contributor, i^* indicates that Private Model i is claimed by a lazy or dishonest computing contributor, $\bar{i}$ indicates that Private Model i from honest computing contributor is under verification and the other models in the model set have been verified, and $\overline{i^*}$ indicates that Private Model i from lazy or dishonest computing contributor is under verification and the other models in the model set have been verified.

As shown in the fifth column in Table IV, the probabilities that verification contributors conclude positive votes for Private Model 1, which is the first claimed private model and generated by the lazy/dishonest Computing Contributor 1, are 10%, 10%, and 10%, respectively, based on the 10 independent simulation trials. We can calculate the probability that the majority voting is positive is 2.8%. Therefore, the probability that this private model can be verified is very low. This result illustrates that our verification mechanism enables our computing paradigm to be robust to the existence of the lazy or dishonest computing contributors. The other results in Table IV are obtained in a similar way, which also illustrates that our computing paradigm is trustworthy even in the presence of lazy or dishonest computing contributor.

D. Case Study III

In this case study, we focus on evaluating the performance of the homomorphic encryption interface in our proposed decentralized and secure computing paradigm by considering a data-driven task on classifying the 10 image classes provided by the MNIST handwritten digit database [50]. The testbed used for this case study is shown

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/TETC.2020.2983007, IEEE Transactions on Emerging Topics in Computing

TABLE II
PARAMETERS OF THE LOCAL MACHINE-LEARNING MODELS CONSIDERED IN CASE STUDY I

CNN parameters			
	Computing Contributor 1	Computing Contributor 2	Computing Contributor 3
Inputs	28x28 images		
Convolution layer 1	32 5x5 kernels	64 5x5 kernels	32 5x5 kernels
Pooling layer 1	2x2 maximum pooling	2x2 maximum pooling	2x2 maximum pooling
Convolution layer 2	16 5x5 kernels	32 5x5 kernels	32 10x10 kernels
Pooling layer 2	2x2 maximum pooling	2x2 maximum pooling	2x2 maximum pooling
Convolution layer 3	8 2x2 kernels	16 2x2 kernels	16 4x4 kernels
Reshaped vector (Convolution layer 3 output are flatten as a vector of size)	7x7x8=392	7x7x16=784	7x7x16=784
hidden layer	Fully connected hidden layer with size 500		
Output	10 labels with softmax activation		
Training method	Adam Optimizer		
Batch size	50		
Learning rate	0.0001		
Maximum number of epochs	100		

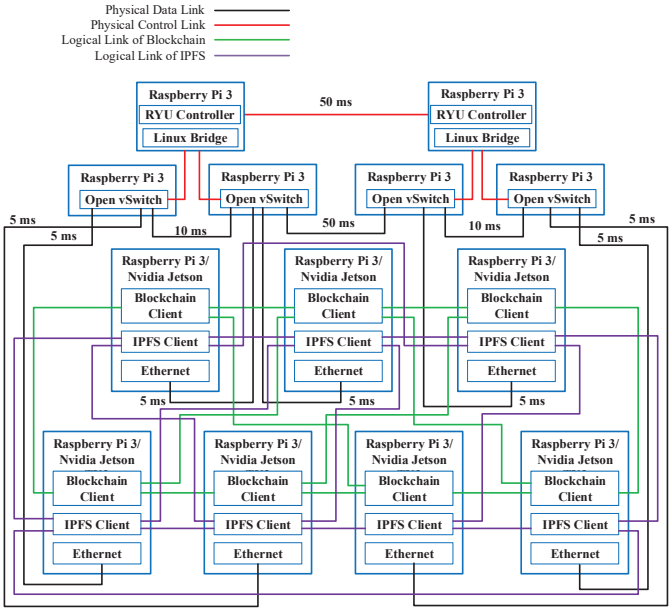


Fig. 10. (a) Illustration of implementing our blockchain-powered and SDN-based decentralized and secure computing testbed.

TABLE III
SUMMARY OF THE DATASET AVAILABLE TO THE INDIVIDUAL CONTRIBUTORS IN CASE STUDY I

Contributor	Set of Labels	No. Training Data	No. Verification Data
Verification Contributors	{0, 1, 2, 3, 4, 5, 6, 7, 8, 9}	1000	1000
Computing Contributor 1	{0, 1, 2, 3, 4}	1000	1000
Computing Contributor 2	{0, 6, 7, 8, 9}	1000	1000
Computing Contributor 3	{5, 6, 7, 8, 9}	1000	1000

in Figs. 9 (a) and 10. The local data owned by the individual computing contributors are detailed in Table V, which shows that the individual computing contributors only have a partial view of the features of the entire dataset. Additionally, we assume that each

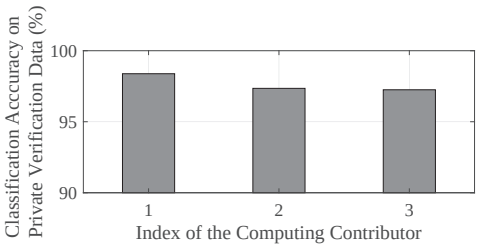


Fig. 11. Accuracy obtained by each computing contributor by using their local verification data.

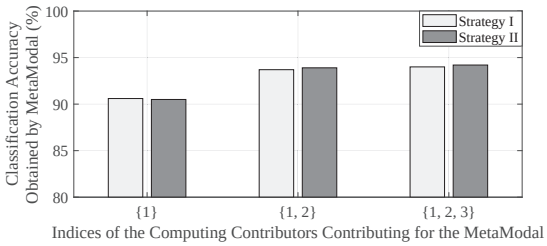


Fig. 12. Comparison of average classification accuracy obtained by Meta-Model using strategies I and II versus the indices of the computing contributors contributing to the MetalModal.

computing contributor locally trains a CNN private learning model with a similar structure as illustrated in Table VI. To perform our homomorphic encryption interface, the floating-point input data and weight parameters of the CNN-based learning models are converted

TABLE IV
SUMMARY OF VERIFICATION DECISION IN CASE STUDY II

Verification Contributor	Probability of Concluding a Positive Vote (%)				
	Models $\{\overline{1}\}$	Models $\{1, \overline{2}\}$	Models $\{1, 2, \overline{3}\}$	Models $\{\overline{1^*}\}$	Models $\{1, 2, \overline{3^*}\}$
1	80	100	80	10	10
2	80	100	80	10	10
3	20	100	80	10	10
Majority Voting	70.4	100	89.6	2.8	2.8

TABLE V
SUMMARY OF THE LOCAL DATA AVAILABLE TO THE INDIVIDUAL CONTRIBUTORS IN CASE STUDY III

Contributor	Set of Labels	No. Training Data	No. Verification Data
Computing Contributor 1	{0, 1, 2, 3, 4, 5, 6}	1000	1000
Computing Contributor 2	{0, 1, 2, 3, 4, 8, 9}	1000	1000
Computing Contributor 3	{0, 1, 2, 6, 7, 8, 9}	1000	1000

TABLE VI
PARAMETERS OF THE CNN-BASED PRIVATE MODELS IN CASE STUDY III

CNN parameters	
Inputs	28x28 images
Convolution layer 1	32 5x5 kernels
Pooling layer 1	2x2 average pooling
Convolution layer 2	16 5x5 kernels
Pooling layer 2	2x2 average pooling
Reshaped vector (Pooling layer 2 output are flatten as a vector of size)	7x7x16=784
Output	10 labels with softmax activation
Training method	Adam Optimizer
Batch size	50
Learning rate	0.0001
Maximum number of epochs	100

to be integer by introducing appropriate scaling factors p and q as shown in Fig 5. In this simulation, we evaluate the performance of the final MetaModel when the computing contributors select the scaling factors $p = 2^5$ or 2^7 and $q = 1000$. Furthermore, we assume that in the encryption interface of the computing contributors, the private CNN-based learning models are encrypted via their first convolution layers. In the encryption interface of the application initiators, the verified private models are decrypted and fused to achieve MetaModels via Fusion Strategy II. The accuracy of the MetaModel achieved with and without homomorphic encryption interface are shown in Table VII, respectively. From Table VII, we can see the MetaModel achieved by using the encryption interface with the scaling factor $p = 2^7$ outperforms that with the scaling factor $p = 2^5$ and achieves comparable accuracy as the original MetaModel. This is reasonable because that the errors caused by rounding the parameters of the CNN-based private models to integers increases when the scaling factor is lower.

E. Case Study IV

In this case study, we focus on comparing the performance of our proposed computing paradigm with that of a widely recognized distributed machine learning paradigm, federated learning, from the perspectives of time complexity and classification accuracy, using a data-driven task on classifying the 10 image classes provided by the MNIST handwritten digit database [50]. In federated learning, the distributed computing contributors initialize their learning models by referring the globally initialized weights maintained in the server agent and then update their own models locally for a certain number of local epochs. The weights of the local models are used to calculate the updated global model weights in the server agent, which completes the first round of model updating. The updated global weights are then distributed to the individual computing contributors for the next round of model-updating procedure. This process is called Federated Averaging (FedAvg).

TABLE VII
COMPARISON OF THE ACCURACY ACHIEVED BY METAMODELS IN DIFFERENT SITUATIONS IN CASE STUDY III

Indices of Fused Models	Classification Accuracy for Fused Models (%)		
	Original	Encrypted ($p = 2^7$)	Encrypted ($p = 2^5$)
{1}	91.5	91.5	89.6
{1, 2}	94.8	94.7	93.0
{1, 2, 3}	95.8	95.8	93.8

To achieve a fair comparison, we assume that the distributed learning is conducted in a secure environment. Under this assumption, the functionality of our homomorphic encryption interface in our computing paradigm is not needed to be considered and only one verification contributor is required. This verification contributor in the context of our proposed computing paradigm is analogous to the server agent in federated learning paradigm. Additionally, our computing paradigm and federated learning have three and four computing contributors, respectively. We consider that federated learning paradigm has one more computing contributor is because that its verification contributor, which is also called server agent, does not conduct any task of model learning while the verification contributor in our proposed paradigm implements model learning. Therefore, to enable a fair comparison environment, we assume there is one more computing contributor in federated learning such that both framework has four nodes with the learning capability.

For federated learning, we assume that the global average process ignores the random selection of local weights and collect all the weights achieved by the available local models. In addition, we assume that each computing contributor adopts the same three-layer CNN models in both of the computing paradigms for comparison. Under this assumption, each computing contributor has the same space complexity when training the private CNN model. The time complexity of a computing paradigm depends on both the training time and the communication cost between the local computing contributors and the verification contributor. Since we consider the same size of training data and the identical training models on the computing contributors in both computing paradigms, it is reasonable to claim that the values of the training time required by the computing contributors belonging to these two computing paradigms are the same. Therefore, the difference between the time complexities of these two computing paradigms depends on the communication complexity. Theoretically, our computing paradigm only needs one round of communication to complete the given data-driven task while federated learning may require multiple communication rounds depending on the given task and training settings. Therefore, generally speaking, the communication cost of our model is $\mathcal{O}(1)$ while federated learning is $\Omega(n)$, where n is the number of communication rounds, which is equal to the ratio between the total epochs and the local epochs. To numerically analyze the time complexities of these two computing paradigm, we measure their running time by utilizing the network simulator NS-3 together with Tensorflow written in Python API. In our simulation, the emulated network has a star topology, where all nodes are connected to a switch. The bandwidth of the network is set to 50 Mbps and propagation delay is 5 ms. Both computing paradigms are running on the same machine with the configuration detailed in Table VIII. To be specific, all the computing and verification contributors are sharing the hardware on a single machine. The communications between the contributors are realized via the emulated channel of NS-3. The total running time includes the computation time of each contributor, synchronization cost, and network delay. The numerical results are shown in Fig. 13.

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/TETC.2020.2983007, IEEE Transactions on Emerging Topics in Computing

TABLE VIII
CONFIGURATION DETAILS OF THE MACHINE RUNNING OUR COMPUTING PARADIGM AND FEDERATED LEARNING

Computer Configuration	
Component	Detail
CPU	Two Intel Xeon 4114 10-core CPUs at 2.20 GHz
RAM	192GB ECC DDR4-2666 Memory
GPU	NVIDIA 12GB PCI P100 GPU
Disk	Intel DC S3500 480 GB 6G SATA SSD
NIC	Dual-port Intel X520-DA2 10Gb NIC

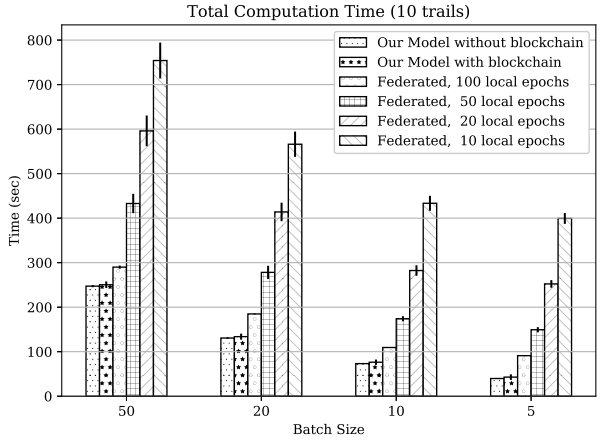


Fig. 13. Performance comparison between our computing paradigm and federated learning from the perspective of time complexity.

To acheive the results, we simulate the model training of 200 epochs. Additionally, in the implementation of our computing paradigm, the computing contributors pre-train their private models for 100 epochs and share the models to the verification contributor that continues to merge the received private model with the fused model by fine tuning the last two layers for another 100 epochs. In the implementation of federated learning paradigm, the number of local epochs is pre-defined, whose value impacts the number of communication round between the computing contributors and the verification contributor. For example, the green bar in Fig. 13 shows that the implementation of the federated learning having 50 local epochs, which implies that the local computing contributors have to synchronize with the verification contributor to update the weights of their private models for $\frac{50}{200} = 4$ round since the total epochs is fixed to be 200 in our numerical analysis. Since federated learning normally needs multiple synchronization rounds while our computing paradigm only requires one round of synchronization process, our computing paradigm outperforms federated learning with significantly reduced time complexity. This is also validated in the numerical results in Fig. 13.

Next, we continue to compare the classification accuracy of our computing paradigm with that of federated learning. The setting remains the same as the one for time-complexity comparison. In federated learning, there are four local computing contributors, each of which involves in five rounds of model updating and conducts 20 local training epoches in each round. Furthermore, we consider two scenarios. In the first scenario, we consider a non-identically independently distributed (non-IID) set of local training samples of the MNIST data available to the computing contributors. Each of these local training samples includes the data associated with all of the 10 image class labels, and the distribution of the local training samples are not identical amongst these computing contributors. In the second scenario, we consider that data are distributed amongst

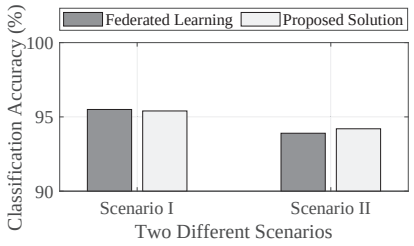


Fig. 14. Performance comparison between our computing paradigm and federated learning from the perspective of classification accuracy.

computing contributors in a way that only part of the 10 image classes are available to most of the computing contributors. The data distribution used in the second scenario is similar to Table III.

The comparison result is shown in Fig. 14, from which it can be seen that both computing paradigms achieve a comparable classification accuracy. In the first scenario where all the contributors have a good view of the entire dataset, federated learning slightly outperforms our computing paradigm. In the second scenario, where most computing contributors only have partial views of the features of the entire dataset, our computing solution is slightly better than federated learning. Furthermore, our computing solution provides a higher integrity security via removing the authoritative controlling agent and introducing the verification contributors. The main functionality of verification contributors is for security purpose rather than for increasing the final accuracy, which also results in the relatively lower accuracy achieved by our solution in the first scenario shown in Fig. 14.

To sum up, our computing paradigm has the comparable performance with federated learning from the perspective of classification accuracy. Furthermore, our paradigm outpeforms federated learning from the perspective of time complexity.

F. Case Study V

In this case study, we evaluate the performance of our decentralized and secure computing paradigm with the functionality of the homomorphic encryption interface in a data-driven task of detecting a fading channel by using link power data. The confidential data used in this task are from Link Power database provided by NASA Glenn Research Center. In the simulation, we assume that one application initiator publishes this data-driven task and there are three computing contributors and three randomly selected verification contributors participating in the task. The link power data are randomly divided into 4 sets, each of which contains 10000 training data and 1000 testing data. Of these four sets, three sets are used as the local data, each of which is available to one of the three computing contributors. The other set is considered to be accessible by being assigned to the three randomly selected verification contributors in the testbed. Additionally, we assume that each computing contributor participating in the task adopts a CNN model with 2 convolution layers and ReLU activation function. In this case study, we consider both Element-wise and Matrix-Pair-wise homomorphic encryption strategies.

To demonstrate the operation of our homomorphic encryption interface, Table IX summarizes the accuracies obtained by each computing contributor with its private model and local testing data with and without the encryption interface. From these results, it is clear that implementing the encryption interface slighted reduces the classification accuracies, which is the cost to achieve higher security and privacy preservation. Additionally, Table X shows the classification accuracy achieved by the MetaModel of the application initiator, which is obtained by fusing the individual verified private

TABLE IX
THE ACCURACY OF FADING CHANNEL DETECTION ACCURACY USING THE PRIVATE CNN-BASED LEARNING MODELS

Model	Detection Accuracy (%)		
	Without Encryption	Element-wise Encryption	Matrix-Pair-wise Encryption
Computing Contributor 1	93	91.6	91.7
Computing Contributor 2	93.1	92.4	92.0
Computing Contributor 3	93.2	92.3	92.1

TABLE X
DETECTION ACCURACY ACHIEVED BY USING METALMODAL

Model	Detection Accuracy (%)	
	Element-wise Encryption	Matrix-Pair-wise Encryption
{1, 2}	94.5	94.5
{1, 2, 3}	95.5	95.0

models via fusion strategy II. From the results in Table X, which is obtained by considering the functionality of homomorphic encryption interface, it is clear that although the final result is satisfactory, the encryption interface slightly reduces the accuracy due to the introduction of randomization as explained in Section V-B and rounding error as discussed in Section VII-D.

Furthermore, we study the impact of the homomorphic encryption interface on the execution time. Tables XI summarizes the overhead for encryption and the time consumed to achieve a local private model by each computing contributor with and without encryption interface. Homomorphic encryption interface is based on Element-wise Encryption. The results are obtained on a Corei5 CPU for 1000 testing data. ML models are executed using TensorFlow. Encryptions and decryptions are performed using Numpy. It can be seen that the execution time with encryption interface is only 2 times higher compared with that without encryption interface.

TABLE XI
THE IMPACT OF THE ELEMENT-WISE ENCRYPTION INTERFACE ON EXECUTION TIME

Model	Execution Time (ms)		
	Overhead for Encryption	Execution without Encryption	Execution with Encryption
Computing Contributor 1	7.7	60	126
Computing Contributor 2	7.6	57	123
Computing Contributor 3	7.6	59	125

VIII. CONCLUSIONS

The availability of sufficient computing power and data are two of the main reasons for the success of machine learning in a variety of application areas. However, both acquisition of computing power and data can be expensive. In many instances, these challenges are addressed by leveraging an outsourced cloud-computing vendor. However, although these commercial cloud vendors provide valuable platforms for data analytics, the performance of these data-driven applications can be mitigated due to a lack of transparency, security, and privacy-preservation. Furthermore, reliance on cloud servers prevents applying big data analytics in the situations where the computing power is scattered. Therefore, there are increasing demands to develop more effective computing paradigms that are effective to process the private and/or scattered data in a suitable decentralized way. To pave the way to achieve this goal, a decentralize, secure, and privacy-preserving computing paradigm is proposed in this paper to enable an asynchronized cooperative computing process amongst distributed and potentially untrustworthy computing nodes that may have limited computing power and computing intelligence. This paradigm is designed by exploring blockchain, decentralized learning, and homomorphic encryption techniques. The performance of the proposed paradigm is evaluated by considering different simulation scenarios and comparing to other related works.

ACKNOWLEDGMENT

This research work was supported by NASA under Grant 80NSSC17K0530. The authors would like to thank Praveen Fernando for assistance with the preliminary work.

REFERENCES

[1] M. van Gerven and S. Bohte, *Artificial neural networks as models of neural information processing*. Frontiers Media SA, 2018.
[2] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
[3] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.
[4] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural networks*, vol. 61, pp. 85–117, 2015.
[5] S. Pandiri, A. Al-Refai, and L. Lundberg, *Cloud Computing - Trends and Performance Issues: Major Cloud Providers, Challenges of Cloud Computing, Load balancing in Clouds*. LAP LAMBERT Academic Publishing, 2012.
[6] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*. ACM, 2015, pp. 1310–1321.
[7] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016.
[8] J. Konečnỳ, H. B. McMahan, D. Ramage, and P. Richtárik, "Federated optimization: Distributed machine learning for on-device intelligence," *arXiv preprint arXiv:1610.02527*, 2016.
[9] B. McMahan and D. Ramage, "Federated learning," https://research.googleblog.com/2017/04/federated-learning-collaborative.html.
[10] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konecny, S. Mazzocchi, H. McMahan, T. Overveldt, D. Petrou, D. Ramage, and J. Roselander, "Towards federated learning at scale: System design," *arXiv preprint arXiv:1902.01046*, 2019.
[11] V. Buterin, "Ethereum: A next generation smart contract & decentralized application platform," *Ethereum White Paper*, 2013.
[12] H. Diedrich, *Ethereum: Blockchains, Digital Assets, Smart Contracts, Decentralized Autonomous Organizations*. CreateSpace Independent Publishing Platform, September 2016.
[13] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system." https://bitcoin.org/bitcoin.pdf, 2008.
[14] V. Vielzeuf, S. Pateux, and F. Jurie, "Temporal multimodal fusion for video emotion classification in the wild," in *Proceedings of the 19th ACM International Conference on Multimodal Interaction*. ACM, 2017, pp. 569–576.
[15] S. E. Kahou, C. Pal, X. Bouthillier, P. Froumenty, Ç. Gülçehre, R. Memisevic, P. Vincent, A. Courville, Y. Bengio, R. C. Ferrari *et al.*, "Combining modality specific deep neural networks for emotion recognition in video," in *Proceedings of the 15th ACM on International conference on multimodal interaction*. ACM, 2013, pp. 543–550.
[16] G. Ye, D. Liu, I.-H. Jhuo, and S.-F. Chang, "Robust late fusion with rank minimization," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 3021–3028.
[17] N. Neverova, C. Wolf, G. Taylor, and F. Nebout, "Moddrop: adaptive multi-modal gesture recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 8, pp. 1692–1706, 2016.
[18] R. L. Rivest, L. Adleman, M. L. Dertouzos *et al.*, "On data banks and privacy homomorphisms," *Foundations of secure computation*, vol. 4, no. 11, pp. 169–180, 1978.
[19] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[20] X. Yi, R. Paulet, and E. Bertino, *Homomorphic Encryption and Applications*. Springer, SpringerBriefs in Computer Science, November 2014.

[21] C. Gentry and D. Boneh, *A fully homomorphic encryption scheme*. Stanford University Stanford, 2009, vol. 20, no. 09.

[22] Y. Masahiro, *Fully Homomorphic Encryption without bootstrapping*. LAP Lambert Academic Publishing, March 2015.

[23] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE transactions on information theory*, vol. 31, no. 4, pp. 469–472, 1985.

[24] H. Zhou and G. Wornell, "Efficient homomorphic encryption on integer vectors and its applications," in *2014 Information Theory and Applications Workshop (ITA)*. IEEE, 2014, pp. 1–9.

[25] USA Patent provisional 62/663,287, 2018.

[26] N. Strom, "Scalable distributed dnn training using commodity gpu cloud computing," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[27] S. Tuli, N. Basumatary, and R. Buyya, "Edgelens: Deep learning based object detection in integrated iot, fog and cloud computing environments," *arXiv preprint arXiv:1906.11056*, 2019.

[28] S. Tuli, R. Mahmud, S. Tuli, and R. Buyya, "Fogbus: A blockchain-based lightweight framework for edge and fog computing," *Journal of Systems and Software*, 2019.

[29] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[30] J. G. Greeno, J. L. Moore, and D. R. Smith, "Transfer of situated learning." 1993.

[31] A.-L. Jin, W. Song, P. Wang, D. Niyato, and P. Ju, "Auction mechanisms toward efficient resource sharing for cloudlets in mobile cloud computing," *IEEE Transactions on Services Computing*, vol. 9, no. 6, pp. 895–909, 2015.

[32] K. Xie, X. Wang, G. Xie, D. Xie, J. Cao, Y. Ji, and J. Wen, "Distributed multi-dimensional pricing for efficient application offloading in mobile cloud computing," *IEEE Transactions on Services Computing*, 2016.

[33] H. Zhang, H. Jiang, B. Li, F. Liu, A. V. Vasilakos, and J. Liu, "A framework for truthful online auctions in cloud computing with heterogeneous user demands," *IEEE Transactions on Computers*, vol. 65, no. 3, pp. 805–818, 2015.

[34] G. Zhou, J. Wu, L. Chen, G. Jiang, and S.-K. Lam, "Efficient three-stage auction schemes for cloudlets deployment in wireless access network," *Wireless Networks*, vol. 25, no. 6, pp. 3335–3349, 2019.

[35] D. G. Roy, P. Das, D. De, and R. Buyya, "QoS-aware secure transaction framework for internet of things using blockchain mechanism," *Journal of Network and Computer Applications*, vol. 144, pp. 59–78, 2019.

[36] A. Wilczyński and J. Kołodziej, "Modelling and simulation of security-aware task scheduling in cloud computing based on blockchain technology," *Simulation Modelling Practice and Theory*, vol. 99, 2020.

[37] Z. Xiong, Y. Zhang, D. Niyato, P. Wang, and Z. Han, "When mobile blockchain meets edge computing," *IEEE Communications Magazine*, vol. 56, no. 8, pp. 33–39, 2018.

[38] A. Stanciu, "Blockchain based distributed control system for edge computing," in *2017 21st International Conference on Control Systems and Computer Science (CSCS)*. IEEE, 2017, pp. 667–671.

[39] B. Zhou, S. N. Srirama, and R. Buyya, "An auction-based incentive mechanism for heterogeneous mobile clouds," *Journal of Systems and Software*, vol. 152, pp. 151–164, 2019.

[40] K. Driscoll, B. Hall, M. Paulitsch, P. Zumsteg, and H. Sivencrona, "The real byzantine generals," in *The 23rd Digital Avionics Systems Conference (IEEE Cat. No. 04CH37576)*, vol. 2. IEEE, 2004, pp. 6–D.

[41] Z. Ghodsi, T. Gu, and S. Garg, "Safetynets: Verifiable execution of deep neural networks on an untrusted cloud," in *Advances in Neural Information Processing Systems*, 2017, pp. 4672–4681.

[42] O. Goldreich, *The Foundations of Cryptography*. Cambridge University Press, 2004, vol. 2.

[43] Agorise, "c-ipfs: IPFS implementation in C. Why C? Think Bitshares' Stealth backups, OpenWrt routers (decentralize the internet/meshnet!), Android TV, decentralized Media, decentralized websites, decent." *Github.com*, October 2017.

[44] E. Heilman, A. Kendler, A. Zohar, and S. Goldberg, "Eclipse attacks on bitcoin's peer-to-peer network," in *24th USENIX Security Symposium (USENIX Security 15)*, November 2015, pp. 129–144.

[45] S. Ethier, *The Doctrine of Chances: Probabilistic Aspects of Gambling (Probability and Its Applications)*. Springer, May 2010.

[46] M. Castro, B. Liskov *et al.*, "Practical byzantine fault tolerance," in *OSDI*, vol. 99, no. 1999, 1999, pp. 173–186.

[47] M. Walfish and A. J. Blumberg, "Verifying computations without reexecuting them," *Communications of the ACM*, vol. 58, no. 2, pp. 74–84, 2015.

[48] D. Liu, "Efficient processing of encrypted data in honest-but-curious clouds," in *2016 IEEE 9th International Conference on Cloud Computing (CLOUD)*. IEEE, 2016, pp. 970–974.

[49] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

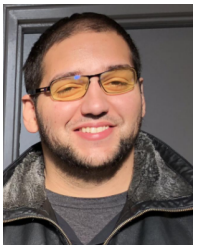
[50] Y. LeCun, "The mnist database of handwritten digits," *http://yann. lecun. com/exdb/mnist/*, 1998.

Gihan J. Mendis is a Ph.D. candidate in Computer and Information Technology at Purdue University. He received the MSc. degree in Electrical Engineering from The University of Akron, in 2016, and he received the BSc. Eng(Hons) degree in Electronics and Telecommunication Engineering at The University of Moratuwa, Sri Lanka, in 2012. His research interests include the deep learning algorithms, decentralize deep learning, encrypted deep learning, cognitive radar sensors, and cognitive radio.

Yifu Wu is a Ph.D. candidate in Computer and Information Technology at Purdue University. He received the M.E. degree in VLSI Systems from the University of Limerick, Ireland, in 2015, and the B.E. degree in automation from the Harbin Institute of Technology, China, in 2011. His research interests include the middleware in computer networking, cyber-physical systems security, blockchain security, deep learning algorithm, and language modeling.

Jin Wei (Kocsis) is an Assistant Professor of Computer and Information Technology (CIT) at Purdue University. She received her Ph.D. degree in Electrical and Computer Engineering from University of Toronto in 2014. Her research interests include developing blockchain-powered decentralized and secure computing paradigm and designing low-complexity machine learning algorithm. Based on the research achievements, she has received multiple research awards including NASA Early Career Faculty Grant and DoE/SuNLaMP Award. She has also achieved multiple best paper awards for the journal and conference publications.

Moein Sabounchi received the B.Sc and M.Sc. degree in electrical engineering from respectively Razi university of Kermanshah and Ferdowsi university of Mashhad in 2011 and 2014. He is currently a Ph.D. candidate at the Department of Computer and Information Technology at Purdue university. His current research interests include the utilization of data-driven algorithms novel technologies in control and signal processing.



Rigoberto Roche' is currently a senior engineer in machine learning and signal/image processing at Lockheed Martin aerospace and defense company. Before joining to Lockheed Martin, he worked as a lead technologist and mission commander at NASA Glenn Research Center. He received the MSc. degrees in Electrical and Electronics Engineering and Biomedical/Medical Engineering at the Florida International University in 2014 and 2013, respectively. He received the BSc. degree in Biomedical Engineering at Florida International University in 2011.