sustainability

MDPI

Article

BlockDeepNet: A Blockchain-Based Secure Deep Learning for IoT Network

Shailendra Rathore [1], Yi Pan [2] and Jong Hyuk Park [1,*]

[1] Department of Computer Science and Engineering, Seoul National University of Science and Technology, Seoul 01811, Korea
[2] Department of Computer Science, Georgia State University, Atlanta, GA 30302-5060, USA
* Correspondence: jhpark1@seoultech.ac.kr; Tel.: +82-2-970-6702

Received: 26 May 2019; Accepted: 18 July 2019; Published: 22 July 2019

Abstract: The recent development in IoT and 5G translates into a significant growth of Big data in 5G—envisioned industrial automation. To support big data analysis, Deep Learning (DL) has been considered the most promising approach in recent years. Note, however, that designing an effective DL paradigm for IoT has certain challenges such as single point of failure, privacy leak of IoT devices, lack of valuable data for DL, and data poisoning attacks. To this end, we present BlockDeepNet, a Blockchain-based secure DL that combines DL and blockchain to support secure collaborative DL in IoT. In BlockDeepNet, collaborative DL is performed at the device level to overcome privacy leak and obtain enough data for DL, whereas blockchain is employed to ensure the confidentiality and integrity of collaborative DL in IoT. The experimental evaluation shows that BlockDeepNet can achieve higher accuracy for DL with acceptable latency and computational overhead of blockchain operation.

Keywords: IoT; deep learning; blockchain; security and privacy; collaborative deep learning

1. Introduction

The rapid advancement in 5G-envisioned industrial automation combines IoT and leading mobile wireless connectivity (5G) to deliver high mobility, availability, and connectivity support in industrial automation. It further enables significant growth in data generation on the IoT platform due to the integration of autonomous machine, mobile robot, and many sensors [1,2]. A recent analyst report valued the global IoT data-management market at around $27.13 billion in 2017 and predicted that it would reach approximately $94.47 billion in 2024, growing at Compound Annual Growth Rate (CAGR) of slightly above 19.51 percent between 2018 and 2024 [3]. To support the analysis of large amounts of data and provide valuable information for the prediction, classification, and detection of future events in IoT, the Deep Learning (DL) paradigm is often employed. As a strong analytic tool, DL enables the reliable mining (feature extraction and representation) of IoT big data generated and collected from various complex and noisy environments [4]. It has become a promising approach for various informatics applications, such as autonomous driving, human activity recognition, anomaly detection, bioinformatics, object detection, and face recognition in IoT [5]. The convergence of DL and IoT has been applied in many real-world tasks and considered the most desirable choice to solve the IoT big data problems. For instance, DL can support the accurate prediction of electricity power consumption in the smart home by collecting and processing data from smart meters, which further help in improving the electricity supply from the smart grid.

With the significant growth in data-driven applications and generated data, however, DL is expected to be more resource-intensive and to be the dominant computing workload in IoT in the future. To support efficient big data analysis and handle dominant computing workload in

IoT, various DL paradigms have been presented in the recent year. The existing paradigms can be divided into three categories: (1) Cloud intelligence, (2) Fog intelligence, and (3) Edge intelligence. In cloud intelligence, a single DL model is deployed at the central cloud server to perform big data analysis in IoT [6]. Note, however, that the IoT network generates a large amount of data that impose heavy communication and computation tasks on the cloud server, such as raw data collection, pre-processing, features extraction, and analysis; hence, the rapid escalation in the server calculation load and communication frequency that further lead to overwhelming, and sometimes unbearable, load to the centralized servers. To address these challenges, Fog intelligence—wherein the data collection and computation tasks of DL are shifted to the fog layer in a distributed manner—was presented [7]. Each fog node prepares a DL model by training its own private data individually and periodically shares the intermediate parameters of the model to the cloud server. The cloud server then aggregates the received parameter and continuously updates the individual model at each fog node. Consequently, the distributed learning at each fog node lowers the computation and communication overhead for the cloud server. Note, however, that the cloud server has full control over the individual learning model, which leads to single point of failure. On the other hand, edge intelligence has the cloud server offloading the intermediate learning and computation tasks to the edge layer. To offload the processing, edge intelligence distributes the DL model in such a way that the lower layers of the model are trained into edge nodes and the higher layers of the model are processed into the cloud server. This is because parts of the DL layers are offloaded at the edge, which, in turn, reduce the intermediate data computation and processing at the cloud layer [8]. Nonetheless, the cloud server in edge intelligence still has full control on the DL model as higher layers are trained at the cloud server, which might result in single point of failure. In addition, all three paradigms do not maintain the full privacy of the data contributors (IoT devices). This is because IoT devices are excluded from the DL process as the data for the learning process are collected without the permission of the device owner, which may cause a privacy leak. Due to the privacy issue, some data contributors do not share their data to the server; hence the difficulty in accumulating enough training data for the distributed DL model, which, in turn, leads to a weak DL model with low accuracy. Furthermore, all three forms of intelligence may suffer from a data poisoning attack wherein any of the entities in the IoT network may demonstrate adversarial behavior in the distributed DL process, e.g., a malicious server can deliberately implant misleading training data, which may disrupt the distributed DL process and wrongly update the model parameters into distributed DL models. Malicious data poisoning at the cloud server may disrupt the individual DL model at the edge of the IoT network [9].

From the aforesaid discussion of existing DL paradigms, we pointed out four major challenges that require solving to design an effective DL paradigm for IoT: (a) Single point of failure, (b) Privacy leak, (c) Training data insufficiency, and (d) Data poisoning attack. Given the migration of the DL platform on the IoT edge device, the learning task is performed directly on the edge devices without depending on the cloud server. Recently, Matt and Rei et al. [10] deployed a DL model directly on an edge device using Google TensorFlow models and Project Flogo. Moreover, Tang, et al. [11] demonstrated that the DL task on an IoT device can benefit from the tailoring of individual device data on the device itself, thus maintaining data privacy. Moreover, data training performs nearer to the device, which reduces the communication overhead and ensures low latency. In this sense, the DL task on an IoT device can be regarded as advantageous in terms of not only offloading computation overhead, low latency, and cost saving, but also as a solution to the problem of single point of failure and privacy leak.

On the other hand, a blockchain supporting a secure network over the untrusted parties has recently become an attractive choice to provide a secure, scalable IoT network [12]. It delivers the silent features of decentralization and immutable distributed ledger to provide privacy in an IoT network. Decentralization in blockchain supports secure sharing of resources and data among all the member nodes in an IoT network that overcomes many-to-one traffic flows and central control dependency and provides enough data for big data analysis. Consequently, it eliminates the problem of single point of failure and training data insufficiency and reduces the communication delay. In addition,

the immutable distributed ledger securely maintains information about all the member nodes of an IoT network in the form of blocks cryptographically controlled by all member nodes. This feature prevents malicious tampering of training data, which, in turn, responds very well to data poisoning attacks in an IoT network.

The DL task on an IoT device (device intelligence) and Blockchain can jointly solve the challenges in the existing DL paradigms for IoT. Therefore, the primary goal of our study is the integration of device intelligence and blockchain technique to support a collaborative, secure DL paradigm for IoT. The research contributions of our study include:

- Introduction of BlockDeepNet, a Blockchain-based secure DL for IoT network that can support a secure, collaborative DL at the device level and which provides data integrity and confidentiality in the IoT network.
- Introduction of a collaborative DL mechanism in the blockchain environment to support local learning models at the device level and aggregation of these local learning models at the edge server through blockchain transactions.
- Presentation of prototype implementation and experimental evaluation to demonstrate the practicability and feasibility of the BlockDeepNet system.

2. Related Work

Along with the recent advancement in the IoT, many researchers have presented their methodologies to provide an efficient big data analysis in IoT. The recent methodologies are mainly focused on the use of DL model that typically contains three layers: input layer, hidden layer, and output layer. A multiple hidden layer (called depth of the model) is used in the DL model, wherein each hidden layer contains a certain number of neurons to provide an accurate feature representation and extraction at the different level. Typically, the neurons at each layer are responsible for the learning of hierarchical features from the input data and give a final output at the output layer in terms classification, prediction, and many more [5]. A neuron takes multiple inputs and gives a single output. The neurons at the l^{th} layer take input from the output of the neurons at the previous $(l-1)^{th}$ layer. To connect the neurons from the subsequent layer, weight and bias parameters are used. For instance, a neuron j at the l^{th} layer relates to a neuron i at the previous $(l-1)^{th}$ layer using model parameters $\left(w_{ij}, b_i\right)$. These model parameters are required to be learned to minimize the model error (i.e., the difference between final model output and targeted value) for accomplishing the DL task successfully. A back-propagation approach that relies on the gradient descent method is used to learn the model parameters efficiently. The detailed process for the back-propagation approach can be found in Abeshu et al. [7].

The abovementioned process to train a multilayer and complex DL model entails a high computational overhead. To mitigate this issue, various novel DL paradigms have been proposed by many researchers at the different layer of IoT such as cloud edge, and fog [4–8]. Zhang et al. [6] presented a DL approach wherein the learning task is performed at the cloud layer to improve the learning efficiency for big data analysis. In Abeshu et al. [7], the authors proposed an innovative distributed deep learning approach called a Fog-to-Things Computing, wherein various fog nodes at the fog layer are responsible for carrying out the DL task to mitigate issue of less scalability and low accuracy in big data analysis. They evaluated their proposed approach using a case study of attack detection in IoT and demonstrated the effectiveness of the distributed DL. Li at al. [8] examined the possibility of DL into the edge computing environment and designed a new offloading methodology to offload the deep learning task from cloud to edge layer in IoT. The authors demonstrated that the DL task can be carried out by training the lower layer of DL at the edge nodes and the higher layer of the DL at the cloud layer. Mohammadi et al. [4] summarized and analyzed existing researches reported of DL in IoT.

As described in the preceding section, the existing research faces major challenges, such as Single point of failure, Privacy leak, Training data insufficiency, and Data poisoning attack. To mitigate these challenges, our research focused on performing the collaborative DL task at the device level with the integration of blockchain technology. In Tang et al. [11], the authors demonstrated that

the DL task at the device level supports protecting the data privacy at the device by performing data analysis on the device itself. On the other hand, our recent research, Rathore et al. [12] provides the integration of Blockchain and IoT to support a secure and scalable IoT network, wherein blockchain can overcome the single point of failure and Training data insufficiency by performing the DL task in a decentralized way. As benefited from the integration of device intelligence and blockchain, BlockDeepNet, a Blockchain-based secure DL for IoT network, is introduced in the subsequent section.

3. System Infrastructure of BlockDeepNet

The system infrastructure proposed in this paper describes a novel decentralized big data analysis approach wherein the learning task is performed at the device level and distributed by employing blockchain technology. A modern IoT network employs three basic steps for big data analysis: a) the data is collected from the IoT device and preprocessed at the connected server; b) The intelligent learning paradigms are used to analyzed the processed data; and, c) with the help of the analyzed data, the IoT device is remotely controlled. A centralized management and control are supported by the existing big data analysis mechanisms; however, an attacker can exploit a significant amount of data maliciously. The proposed BlockDeepNet system remarkably lowers the possibility of data being manipulated adversely by facilitating a secure and collaborative DL paradigm. Consequently, some components of the IoT network must be reconfigured to support the working procedure of BlockDeepNet.

3.1. Reconfigured IoT Network

A reconfigured IoT network is employed to carry out a collaborative DL task. A layered architecture of the IoT network is general as in Li et al. [8], but with a distinct device and edge layer, as shown in Figure 1. Each IoT device at the device layer is configured with blockchain application and the DL model. An edge server at the edge layer is reconfigured for a collaborative DL and blockchain mining task. The operations and interaction in the device and edge layer are facilitated through the transaction implemented using blockchain technology in BlockDeepNet, which is isolated or independent from the central cloud server. In the reconfigured IoT network, each device performs the DL task by collecting and processing their private data.

3.2. Functional Module of BlockDeepNet

In this subsection, we describe the BlockDeepNet system and its functional components. Figure 2 shows an overview of the system design, wherein IoT devices interact with their associated edge server through blockchain to carry out a collaborative and secure DL task. Along the path between IoT devices and the edge server, six key components of BlockDeepNet, as depicted in Figure 3, are described as follows:

1) Local model: Each IoT device prepares a local learning model by employing DL over its own private data.
2) Smart Contract: The smart contract defines all the policies and rules for operating and governing the BlockDeepNet system. It consists of two main modules: learning and mining contracts. Through learning contract, each IoT device sends out the parameters of its local model referred to as local update to the blockchain network. The edge server downloads the learning contract for further processing wherein a collaborative DL model is trained by the edge server with the help of a local model update shared through its associated IoT devices. Finally, the parameters of the collaborative DL model referred to as global update are mined by the edge server and sent out to the blockchain network, which is called the mining contract. The corresponding global update is downloaded by each IoT device, and their local model is updated.
3) Smart contract interface: It connects an IoT device and the edge server to the smart contract. It automatically triggers smart contract operations and activities at the IoT device layer, such as IoT

device registration, communication among IoT devices, sharing of local models of the IoT devices with the edge server, and IoT device requesting for collaborative DL model. In the proposed BlockDeepNet system, we employed a JavaScript-based Application Programming Interface (API) called Web3 protocol to deploy the smart contract interfaces for IoT devices. For each IoT device, programming functions are called by a smart contract interface to execute the rules defined in the smart contract.

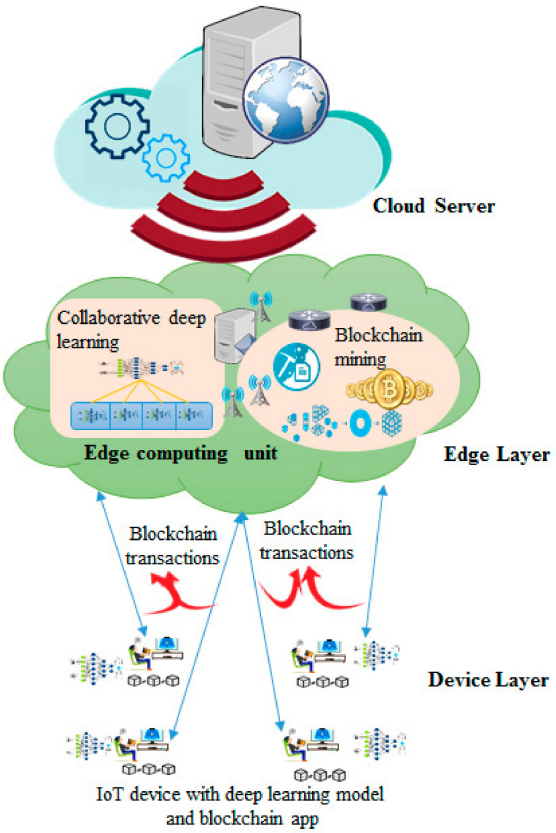


Figure 1. Reconfigured IoT network for deep learning.

4) Blockchain network: The BlockDeepNet system employs a private blockchain to deliver a collaborative DL task in IoT, wherein an edge server has more control on the blockchain network. We chose private blockchain since a very resource-intensive proof of work is not needed for consensus and there is less possibility of sybil attacks. Moreover, a mining task does not require an economic incentive, and a less resource-intensive mining task is provided compared to that of a public blockchain. In this sense, since IoT devices are resource constraints, they serve as blockchain and smart contract clients in our system, and a mining task is carried out only by the edge server in the blockchain network of the BlockDeepNet system. In other words, the entire task of transaction monitoring, new block creation, and propagation is done by a resource-intensive edge server. The IoT devices interact with the edge server by installing blockchain and smart contract software (blockchain application), and they are able to obtain resources and support for offloading their processing job using mechanisms such as cloud offloading. In the BlockDeepNet implementation, the deployment and distribution of a smart contract are carried out on the blockchain. The blockchain service is supported by a blockchain server where IoT devices are connected to it as a client. The blockchain server performs two key operations to support the blockchain service. First, the server collects all the transactions among the IoT devices and runs the smart contracts. It generates new blocks to support the execution of embedded code in the smart contracts. Second, the blockchain

server records all the activities in the system such as information about requesting and logging devices and mining blocks. Note that the blockchain server in the BlockDeepNet system uses a lightweight consensus mechanism such as PBFT [13], which does not require proof-of-work task and supports less resource-intensive mining tasks [14].

5) Iteration and epoch: In the BlockDeepNet system, iterations refer to the multiple steps of DL tasks performed by an IoT device to obtain an effective local model. On the other hand, an epoch refers to a single operation of generating a new candidate block in the blockchain by the edge server.

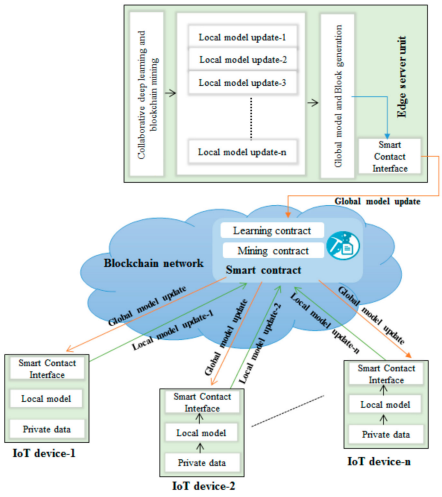


Figure 2. Design overview of proposed BlockDeepNet system.

4. Working Mechanism of BlockDeepNet

In order to resolve the issues in the existing works, we introduce a BlockDeepNet system wherein the blockchain network is leveraged to support a collaborative and secure DL task in IoT. The conceptual structure of the BlockDeepNet system consists of edge server and IoT devices. An edge server can be associated with a set of IoT devices, working as a miner in the private blockchain network. The working mechanism of the BlockDeepNet system is summarized as follows. Each IoT device in the BlockDeepNet system trains a local model using DL and uploads the local update (gradient parameters, learning model's weight) to its associated edge server in the blockchain network. In return, the edge server pays data reward to the IoT device based on the amount of sample data it has. The edge server verifies and processes the local updates of its associated devices to generate a collaborative DL model by aggregating local updates. Once the edge server accomplishes the collaborative DL task, it creates a block wherein the aggregated local updates are stored and receives the processing reward from the blockchain network. Finally, the created block is added to the blockchain network, which is also called a distribute ledger. Each IoT device downloads the distributed ledger and updates its local update to compute a global update, which is an input of the next local update. Since an IoT device locally computes its global update, malfunctioning of the edge server or another IoT device during the learning process does not affect the device global update, thus overcoming single point of failure and ensuring the security and robustness of the overall DL task in IoT. Based on these benefits, in order to design the BlockDeepNet system, we present in this section a more detailed explanation about the collaborative DL process and its deployment in a blockchain environment. We also discuss the latency delay invoked by the collaborative DL and blockchain operation.

4.1. Collaborative DL Process

The DL task in IoT relies on a collection of parameters (learning model's weight w) obtained by learning on given training data. A sample of training data i is described as a two-dimensional coordinate $\left(x_i, y_i\right)$, wherein the DL model takes vector x_i as an input (such as pixels of an image)

and gives scalar output y_i (such as the label of the image). For each training data sample, a DL model computes a loss function defined on its parameter vector w to support the learning process. The loss function provides the DL model's error on the training data, which is minimized in the learning process by minimizing the loss function on a set of training data sample [15]. The loss function for data sample i can be defined as $f\left(w; x_i; y_i\right)$ and written as $f_i(w)$.

Assume that a set of IoT device $d = \{d_1, d_2, \ldots, d_k, \ldots, d_n\}$, where device d_k has a collection of data samples s_k and it is associated with edge server e. A loss function for device d_k over data samples s_k can be defined as $\mathcal{F}_k(w) \triangleq \frac{1}{|s_k|} \sum_{i \in s_k} f_i(w)$.

We consider $s_k \triangleq |s_k|$, where $|.|$ stands for the size of data sample s_k and $s \triangleq \sum_{k=1}^{n} s_k$ denotes the size of total data samples for all devices. Assuming $s_k \cap s_{k'} = O$ for $k \neq k'$, we define a global function on all devices associated with edge server e as:

$$\mathcal{F}(w) \triangleq \frac{\sum_{k=1}^{n} \mathcal{F}_k(w)}{s} \tag{1}$$

The DL task is to find a minimized $\mathcal{F}(w)$, which can be defined as follows:

$$w^* \triangleq \arg\min \mathcal{F}(w) \tag{2}$$

Note, however, that finding a closed-form solution for Equation (2) is usually impossible because of the inherent complexity of the DL model.

To solve Equation (2) in the BlockDeepNet system, we present a canonical collaborative DL paradigm inspired by a federated learning system (e.g., [16]). Local model parameters $w_k(t)$ are defined by each device d_k by training its local DL model over data sample s_k. Here, $t = 0, 1, 2, \ldots, T$ represents the iteration index. Initially, all devices d_k in d set the value of their local model parameters $w_k(t)$ to the same value at $t = 0$. At $t > 0$, each device computes a new value of $w_k(t)$ based on the parameter value in the previous iteration $(t-1)$ by minimizing the local loss function using a gradient-descent update rule [16,17]. This minimization step of local loss function using the gradient-descent update rule at each device d_k is termed local update. Subsequently, global aggregation is carried out after one or multiple iterations of local update, wherein the weighted average is calculated for the local update of all device n termed global update, which is further used to update local update at each device d_k. We define each iteration to be inclusive of a local update step that is possibly followed by a global aggregation step.

The local update rule for device d_k is defined as follows: $w_k^{(t)} = w_k^{(t-1)} - \lambda \nabla \mathcal{F}_k\left(w_k^{(t-1)}\right)$, where $\lambda > 0$ is a step size. We define the weighted average of $w_k^{(t)}$ for a set of all device d at any local iteration t (that might or might not involve a global aggregation stage) as below: $w^{(t)} = \frac{\sum_{k=1}^{n} s_k w_k^{(t)}}{s}$.

To elaborate the global update, we define global update to be updated up to P epochs. For each epoch, a total of $\mathcal{T}$ iterations of the local update are performed at each device d_k. The local update for device d_k at the t^{th} iteration of the p^{th} epoch is denoted by $w_k^{(t,p)}$ and defined as follows:

$$w_k^{(t,p)} = w_k^{(t-1,\,p\,)} - \frac{\lambda}{s_k}\left(\left[\nabla \mathcal{F}_k\left[w_k^{[t-1,\,p\,]}\right] - \nabla \mathcal{F}_k\left[w_k^{[p-1]}\right]\right] + \nabla \mathcal{F}\left(w^{(p-1)}\right)\right) \tag{3}$$

where, at the p^{th} epoch, the global update is defined by $w^{(p)}$, and $\nabla \mathcal{F}\left(w^{(p)}\right) = 1/s \cdot \sum_{k=1}^{n} s_k \nabla \mathcal{F}_k\left(w^{(p)}\right)$ is found using Equation (1). Let $w_k^{(p)}$ indicate the local update at device d_k after the last local iteration of the p^{th} epoch such that $w_k^{(p)} = w_k^{(\mathcal{T},\,p)}$. Then, we update the global weights at the p^{th} epoch $(w^{(p)})$ as follows:

$$w^{(p)} = w^{(p-1)} + \frac{1}{s} \sum_{k=1}^{n} s_k \left(w_k^{(p)} - w^{(p-1)}\right) \tag{4}$$

The process of local updates and global aggregations is continued until the following condition is satisfied: $\left|w^{p} - w^{p-1}\right| \leq \varepsilon$ for constant $\varepsilon > 0$. The logic of a collaborative DL paradigm is shown in Algorithm 1, where the $\mathcal{T}$ iterations of the local update are performed in epoch p at each device d_k. For ease of presentation, each iteration t is checked as an integer multiple of $\mathcal{T}$ to define the global aggregation in a theoretical analysis. It should be noted that the communication aspect between IoT devices and edge server is not considered in Algorithm 1. We will discuss such aspect in the subsequent subsection. Algorithm 1 gives w_f as the final model update or parameter that produces a minimum value of global loss over an entire execution of local and global updates.

4.2. Blockchain in Collaborative DL

In order to provide a secure and reliable exchange of local and global updates among edge server and IoT devices via the distributed ledger, the BlockDeepNet system deploys collaborative DL in the blockchain network, wherein blocks and their verification are carried by the edge server (miner). Like the traditional blockchain, in the distributed ledger of BlockDeepNet, each block contains its header and body [18]. The body consists of a set of verified transactions $Tx = \left\{Tx_{d_1}^{\{p\}}, Tx_{d_2}^{\{p\}}, \ldots, Tx_{d_k}^{\{p\}}, \ldots, Tx_{d_n}^{\{p\}}\right\}$, wherein each transaction $Tx_{d_k}^{(p)}$ envelops with a local update from device d_k at epoch p and corresponding computation time of local update $T_{local,\, d_k}^{(p)}$. Note that the local update is $E\left(w_k^{(p)}\right)$ for device d_k at epoch p, where local weight $w_k^{(p)}$ is encrypted with encryption E. The header encompasses information about block generation rate β and pointer to previous block $\aleph$. Thus, the size of a block can be defined as $h + \mathcal{S} \times n$ with given header size h and average size of local update $\mathcal{S}$ for each device in n.

Algorithm 1: Collaborative DL Paradigm.

Input: $d = \{d_1, d_2, \ldots, d_k, \ldots, d_n\}$: a set of IoT devices associated with edge server e, s_k: a set of data samples having a device d_k, $\mathcal{T}$: total number of iterations in an epoch, P: total number of epochs, ε: threshold error.
Output: Final model update or parameter $w^{(f)}$
Process:
Initialize: local iteration $t = 0$, epoch $p = 1$, model parameter $w^{(f)}$, $w_k^{(0,1)}$, $w^{(0)}$, $w_k^{(0)}$ to the same value for all device d_k.
while $\left(\left|w^{|f|} - w^{|p-1|}\right| \not\leq \varepsilon\right)$ **do**
{
Set $t \leftarrow t + 1$
For each device d_k in the d **do**
    Compute local update $w_k^{(t,p)}$ using Equation (3)
If t is an integer multiple of $\mathcal{T}$
    Compute global update $w^{(p)}$ using Equation (4)
    Set $w_k^{(p)} \leftarrow w^{(p)}$
    Update $w^f \leftarrow \text{argmin}_{w \in \{w^f,\, w^{|p|}\}} \mathcal{F}(w)$
    Set $p \leftarrow p + 1$
Else
Set $w_k^{(t,p)} \leftarrow w_k^{(t,\ p)}$
}
End

The BlockDeepNet system initializes with a genesis block generation phase, assuming that each IoT device has been registered in the blockchain network where address p_k corresponding to the blockchain server is assigned to each device to launch a transaction. The edge server generates a genesis block

through genesis block generation, which contains initial transactions recording ownership statements for each device. After the genesis block is created, the edge server generates a candidate block at each epoch p that is filled with a set of verified transactions (local updates) from its associated devices. Algorithm 2 describes the operational steps to generate a candidate block at the p^{th} epoch in the BlockDeepNet system, where a collaborative DL as described in Algorithm 1 is deployed in a blockchain environment. In Algorithm 2, each device d_k initializes model parameters and computes their local update by invoking Algorithm 1.

Algorithm 2: A Candidate Block Generation Process in BlockDeepNet

1: **Input:** As described in Algorithm 1
2: **Output:** global update $w^{(p)}$ for a device d_k
3: **Initialize (for epoch p = 1)**: i.e., Line 4 in the Algorithm 1.
4: **Compute** local update $\{w_1^{(p)}, w_2^{(p)}, \ldots, w_k^{(p)}, \ldots, w_n^{(p)}\}$, i.e., Line 7 to 9 in the Algorithm 1.
5: **Learning Contract:** IoT Devices
Key generation:
6: Public key (PK): $\left(pk_{d_1}^{psu}, pk_{d_2}^{psu}, \ldots, sk_{d_k}^{psu}, \ldots, pk_{d_n}^{psu}\right)$,
Secret key (SK): $\left(sk_{d_1}^{psu}, sk_{d_2}^{psu}, \ldots, sk_{d_k}^{psu}, \ldots, sk_{d_n}^{psu}\right)$.
7: **For each** device d_k in the d **do**
8: **Compute** $E(w_k^{(p)}) = Encryption\left(w_k^{(p)}, pk_{d_k}^{psu}\right)$
9: **Envelope** $Tx_{d_k}^{(p)} \leftarrow [E(w_k^{(p)}), T_{local,\, d_k}^{(p)}]$
10: **Upload** $Tx_{d_k}^{(p)}$ to the edge server
11: **End**
12: **Mining Contract:** Edge server
13: **Receive** $Tx = \left\{Tx_{d_1}^{\{p\}}, Tx_{d_2}^{\{p\}}, \ldots, Tx_{d_k}^{\{p\}}, \ldots, Tx_{d_n}^{\{p\}}\right\}$
14: **For each** $Tx_{d_k}^{(p)}$ in Tx **do**
15: **Extract** $E(w_k^{(p)})$, and $T_{local,\, d_k}^{(p)}$ from $Tx_{d_k}^{(p)}$
16: **Verify** $E(w_k^{(p)})$
17: **End**
18: **Compute** weighted average $w^{(p)}$ using Equation (4)
19: **Envelope** $Tx_{co}^{(p)} \leftarrow E[w^{(p)}]$
20: **Candidate Block B** $\leftarrow [Tx_{co}^{(p)}, \beta, \aleph]$
21: **Learning Contract:** IoT Devices /* Block propagation*/
22: **For each** device d_k in the d **do**
23: **Download** $Tx_{co}^{(p)}$ from the edge server
24: **Extract** $E[w^{(p)}]$ from $Tx_{co}^{(p)}$
25: **Compute** $w^{(p)} = Decryption\left(E\left(w^{(p)}\right), SK\right)$
26: **Update** $w_k^{(p)} \leftarrow w^{(p)}$ /* Compute global update */
27: **End**

A learning contract is then executed by the associated edge server through which each device iteratively uploads its local update to the edge server. In the learning contract, each device d_k trades its local update to the edge server. In the learning contract, each device d_k trades its local update $w_k^{(p)}$ in encrypted form and computation time of local update $T_{local,\, d_k}^{(p)}$ at each epoch p, which are enveloped with transaction $Tx_{d_k}^{(p)}$ and sent to the edge server. An additively homomorphic encryption [19] is used to encrypt the local update, where all devices associated with the edge server generate pseudo public keys $\left(pk_{d_1}^{psu}, pk_{d_2}^{psu}, \ldots, pk_{d_k}^{psu}, \ldots, pk_{d_n}^{psu}\right)$ and cooperatively create the corresponding pseudo

secret keys $SK = \left(sk_{d_1}^{psu}, sk_{d_2}^{psu}, \ldots, sk_{d_k}^{psu}, \ldots, sk_{d_n}^{psu}\right)$, such that $SK = f\left(sk_{d_1}^{psu} + sk_{d_2}^{psu} +, \ldots, +sk_{d_n}^{psu}\right)$, where f is a function of secret sharing protocol. Each device d_k has a fraction of secret key $sk_{d_k}^{psu}$. After receiving transactions $\left\{Tx_{d_1}^{p}, Tx_{d_2}^{p}, \ldots, Tx_{d_k}^{p}, \ldots, Tx_{d_n}^{p}\right\}$ from all associated devices, the edge server verifies and processes these transactions through mining contract and creates transaction Tx_{co}^{p}, which is known as global update at epoch p. Specially, edge server computes the weighted average of $\left(E(w_1^{(p)}\right), E(w_2^{(p)}), \ldots, E(w_n^{(p)}))$ with the help of Equation (4) and properties of additively homomorphic encryption [16] as follows:

$$E[w^{(p)}] = E[w^{]p-1]}\Big] \times \frac{1}{s} \prod_{k=1}^{n} E\Big[w_k^{[p]}\Big]^{s_k} \times E\Big[-w^{[p-1]}\Big]^{s_k} \quad (5)$$

Here, the mining contract is responsible for verifying and processing the local updates from all devices associated with the edge server for computing the global update and creating candidate block $[Tx_{co}^{p}, \beta, \aleph]$, where β and $\aleph$ are the block generation rate and pointer to the previous block, respectively. Through the learning contract, each device d_k downloads the candidate block and computes a global update by decrypting the $E\left(w^{(p)}\right)$ stored in the candidate block. Thus, learning and mining contracts are iteratively executed to carry out the collaborative DL process at each epoch p.

The learning and mining contracts provide two security requisites: confidentiality and secret sharing. In terms of confidentiality, a local update by each device is encrypted with additively homomorphic encryption; therefore, if a device does not reveal its local update, then no one can obtain information about the local update. Moreover, since the edge server only performs computation over encrypted data, it does not reveal what it knows. On the other hand, in secret sharing, each device participating in the collaborative DL process is required to decrypt the global update using the cooperatively created secret key SK.

5. Experimental Analysis

In this section, we present an experimental evaluation of the BlockDeepNet system to show its practicability and feasibility in IoT. First, an experimental testbed is described to implement a BlockDeepNet prototype. An application of object detection is then deployed on the BlockDeepNet system to analyze the compatibility of collaborative DL and IoT. In the third part, we show a security analysis of BlockDeepNet, including how it overcomes the security issues in the existing centralized and distributed system. Finally, a feasibility analysis is presented to validate the capabilities of DL and blockchain operations in the BlockDeepNet system with an acceptable performance overhead.

BlockDeepNet Testbed Setup

The BlockDeepNet testbed setup involves three major components of IoT devices: edge cloud, cluster, and access point. The IoT devices are operated as a front end consisting of several Raspberry Pi 3 Model B single-board computers with Raspbian Operating System (OS), 32GB storage, 1 GB RAM, 1.2GHz CPU, and several accessories such as Google bonnet, microphone, sense hat, and cameras and so on. The front end also includes a laptop with MacOS, 256 GB storage, 4 GB RAM, 2.2 GHz CPU. An OpenStack deployment is configured as a back-end edge cloud cluster consisting of 1 high-performance Cisco 3850 switch, 1 high-performance Dell PowerEdge C730x rack server, and 4 high-performance Dell PowerEdge R630 rack servers. Each OpenStack rack server is configured with 256 GB RAM, 18 independent CPU cores. A CentOS 7 is installed in the edge server as an operating system. A high-performance Cisco WiFi is used as an access point.

Go-ethereum was used to set up the blockchain platform and solidity language was employed to write smart contracts that include two main modules of learning and mining contracts and deployed

using a Truffle development suite. The learning contracts contain the information about the uploading operations (i.e., sending local updates from IoT devices to the rack server) and downloading operations (i.e., getting global updates by IoT devices from the rack server). On the other hand, mining contracts hold the information about candidate block generation and mining task, i.e., computing a Proof-of-Work (PoW). The rack server processes the mining contracts to candidate block generation and mining. Additionally, the interaction between IoT applications and blockchain was supported by employing Node.js as an interface. Subsequently, python version 3.6.4 and Tensorflow version 1.7.0 were configured for a DL operation. The edge server was configured for Go-ethereum. In addition, each Raspberry Pi was also installed with Go-ethereum and DL. The Go-ethereum in a Raspberry Pi was configured such that it operates in light mode without the block mining function. We set up a DL in a raspberry Pi using five steps: 1) Update the Raspberry Pi; 2) Install TensorFlow; 3) Install OpenCV; 4) Compile and Install a Protobuf (Protocol buffers); 5) Set up a TensorFlow directory structure. Here, OpenCV was installed to improve the computational efficiency and support real-time application in IoT. Protobuf was compiled to provide efficient and fast structuring of data for DL.

In the experiment setup, the Raspberry Pis and the laptop were operated as a blockchain and DL clients responsible for generating and sending the local updates to the rack server in the form of blockchain transactions and receiving global updates from the rack server via learning contracts. On the other hand, a rack server acted as a candidate block generator and the block miner solving Proof-of-Work (PoW) for collaborative DL task. Since the edge server records transactions, and processing and mining the blocks via mining contracts, it acts as a full blockchain node. While IoT devices only record their individual transactions and no block mining, they act as a light blockchain node.

In order to validate the compatibility of the BlockDeepNet system in IoT, we evaluated the performance of an object detection task in the IoT platform using the BlockDeepNet system. Object detection is widely used in IoT such as door guarding, crowd control, and city surveillance, requiring low latency and higher accuracy. For object detection, the well-known PASCAL VOC 2012 dataset [20] consisting of 27,450 trainval instances and 13,841 validation instances of 20 object classes was selected. The whole dataset was partitioned into 10 subsets, where the size of a subset was 2,745 (i.e., 27,450/10). The number of subsets depends on the number of Raspberry Pis. In our experiment, we considered a maximum of 10 Raspberry Pis (R-1, R-2, R-3, R-4, R-5, R-6, R-7, R-8, R-9, R-10), and each one of them was assigned to a distinct subset from among 10 subsets. Each raspberry Pi prepared a trained model using a Convolution Neural Network (CNN) with the structure Input $\rightarrow$ Conv $\rightarrow$ Maxpool $\rightarrow$ Fully Connected $\rightarrow$ Output, where the weights and bias parameters are $w_1 = (10, 1, 3, 3)$ and $b_1 = (10, 1)$ for Conv layer, $w_2 = (1960, 128)$ and $b_2 = (1, 128)$ for fully connected layer, and $w_2 = (128, 10)$ and $b_2 = (1, 10)$ for output layer. The value of other training parameters (Learning rate, No. of epochs, No. of iterations) was set to (0.5, 1, 1500).

We implemented the Threshold Paillier algorithm [21] on each Raspberry Pi with 160-line codes in JAVA to carry out the task of additively homomorphic encryption. To support the object detection, a camera module inbuilt with each Raspberry Pi captured video frames (objects) in 1080 p resolution and transferred them to BlockDeepNet system for further processing, wherein detection results were depicted as the boxes over the identified objects. In order to evaluate the performance of object detection, four standard measures: accuracy, security analysis, time delay, and computational complexity were used, which is described as follows:

Mean Precision Accuracy (mPA): Figure 3a,b show the mPA of the object detection task using the BlockDeepNet system in the case of 5 and 10 Raspberry Pis, respectively. We observed that mPA increases in proportion to the number of Raspberry Pis (IoT devices). Generally, all Raspberry Pis prepare their individual local model by training their own private data (subset of 2745 instances) and share their local updates for collaborative DL in the BlockDeepNet system that generates a global update by the weighted average of all local updates. It is obvious that, when a large number of Raspberry Pis participate in BlockDeepNet, the size of the training dataset will increase (5×2745 for 5 Raspberry Pis and 10×2745 for 10 Raspberry Pis), and global update will be obtained from

the larger number of local updates that will further increase the overall mPA. Each Raspberry Pi will get a global update with an overall higher mPA and overcome the issue of lack of training dataset. We also compared the mPA of each Raspberry Pi in case of individual local update (i.e., without BlockDeepNet) and aggregated global update (i.e., with BlockDeepNet). As shown in Figure 3a,b, for all Raspberry Pis, individual local updates (red bars) provide less accurate results compared to that of global update (light green bars) in the BlockDeepNet system. It demonstrates that BlockDeepNet supports higher mPA compared to individual local update.

Latency: We determined the latency delay for object detection using the BlockDeepNet system. Latency was measured as the total time cost for one epoch operation in BlockDeepNet. We evaluated latency with an increasing number of Raspberry Pis (from R-5 to R-10) as shown in Figure 4a. Notable is the fact that latency increases inconspicuously with an increasing number of Raspberry Pis.

Table 1. Qualitative analysis of the proposed BlockDeepNet.

Research Challenges	Without BlockDeepNet	With BlockDeepNet
Single point of failure	Leverages on centralized architecture, wherein a centralized server has full control over DL operations (i.e., raw data collection, pre-processing, features extraction, and analysis) that produces overwhelming, sometimes unbearable, load to the single centralized servers and results in single-point failure.	Employs blockchain technology to provide a secure sharing of resources and data among all the member nodes in the IoT network that distributes load of the DL operation in a decentralized manner and overcomes a central control dependency.
Privacy leak	Unable to preserve the full privacy of the data contributors (IoT devices), since the IoT devices are excluded from DL process as the data for learning process are collected without the permission of device owners, which may cause a privacy leak.	Each IoT device participates in the DL operation and delivers a local model update by employing the DL over its own private data that protect IoT device from a significant privacy leak.
Training data insufficiency	Due to the privacy issue, some data contributors do not share their data to the server resulting in difficulty in accumulating enough training data for the distributed DL model that lead to a weak DL model with low accuracy.	Instead of sharing the data to the server, each IoT device shares the local update (gradient parameters, learning model's weight) of its local model that overcomes the issue of training data insufficiency and provides an accurate global model to each IoT device.
Data poisoning attack	Any of the entities in the IoT network may pose adversarial behavior in the distributed DL process, e.g., a malicious server can implant misleading training data deliberately that may disrupt the distributed DL process and wrongly update the model parameters into distributed DL models.	Each entity in the IoT network communicates securely via blockchain transaction, where the learning and mining contracts provide two security requisites, i.e., confidentiality and secret sharing with the help of additively homomorphic encryption and decryption.

Privacy and security analysis: The BlockDeepNet system supports privacy protection as an important factor. To measure privacy leak, we define a data similarity index wherein the degree of similarity between the reconstruction of an update from encrypted update and actual update is determined. The data similarity can be calculated using many ways—for example, Euclidean distance, etc. As shown in Figure 4, we measured the data similarity index in our experiment in terms of calculation level, which estimates that similarity decreases and privacy increases with increasing level of calculation.

Computational Complexity: The computational feasibility of the BlockDeepNet system was evaluated based on two primary operations of DL and blockchain and found additional computation overhead due to collaborative DL and block generation. The overall overhead of BlockDeepNet can be categorized

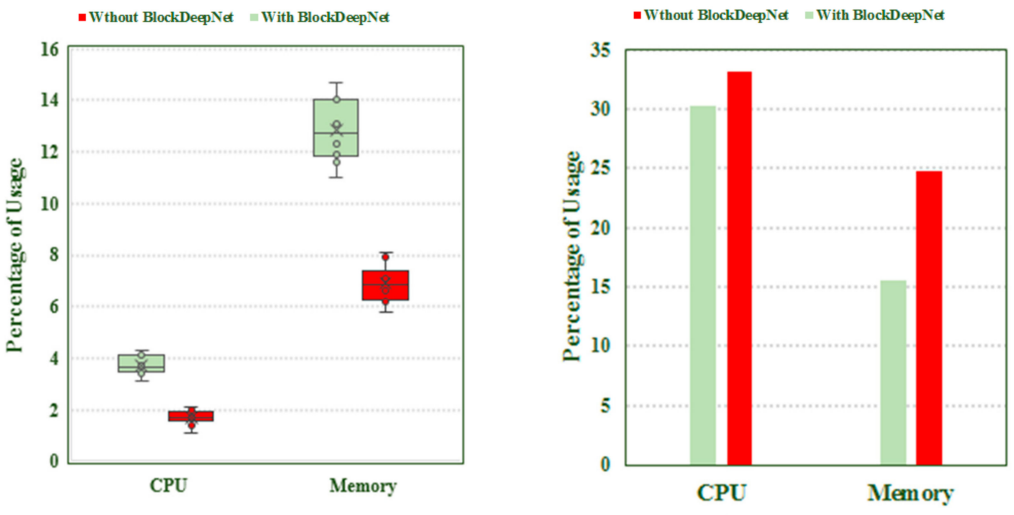
into two aspects: CPU and memory. Figure 5a,b show the overall overhead with and without the BlockDeepNet system for the edge server and IoT devices, respectively. From box plots in Figure 5a, we can see that, with BlockDeepNet, additional CPU (lies between 3.1 and 4.3) and memory (lies between 11 to 14.7) resources are utilized to carry out blockchain and DL operations.

(**a**) 5 Raspberry Pis (**b**) 10 Raspberry Pis

Figure 3. Mean precision accuracy of BlockDeepNet.

(**a**) Latency analysis (**b**) Privacy and security analysis

Figure 4. Performance analysis of BlockDeepNet.



(**a**) IoT device (**b**) Edge server

Figure 5. CPU and memory utilization in BlockDeepNet.

On the other hand, in the case without the BlockDeepNet system, each device sends its private data to the edge server for DL operation that eliminates blockchain and DL overhead from IoT devices and lowers the CPU (lies between 1.1 and 2.1) and memory (lies between 5.8 and 8.1). It further leads to higher CPU and memory without BlockDeepNet compared to the case with BlockDeepNet.

Table 1 summarizes the advantages of BlockDeepNet system over the existing researches based on the four research challenges: Single point of failure, privacy leak, training data insufficiency, and data poisoning attack.

6. Conclusions

In this paper, we proposed BlockDeepNet, a Blockchain-based secure DL system by combining DL and blockchain. BlockDeepNet provides three contributions in the area of DL for big data analysis in IoT. First, a collaborative DL paradigm that supports DL at the IoT device level to mitigate privacy leak and obtains enough data for DL was presented. Second, collaborative DL was deployed in a blockchain environment to provide secure and reliable exchange of local and global updates. Finally, a protype model of BlockDeepNet was developed to validate its effectiveness in real-time scenarios. We conducted an experimental evaluation of BlockDeepNet with a case study of object detection to demonstrate its feasibility and compatibility in IoT. The evaluation results demonstrated that BlockDeepNet is feasible for big data analysis task in IoT such as object detection and provides an efficiency in terms of accuracy, security analysis, time delay, and computational complexity. Our findings suggest that BlockDeepNet mitigates the existing challenges and obtains higher accuracy with acceptable latency and computational overhead of blockchain operation for DL in IoT. However, DL at the device level can further result in the requirement of higher computation power and devices with lower computation power cannot be benefited with BlockDeepNet. To address this issue, BlockDeepNet can be enhanced with an offloading mechanism wherein devices with low computation power can offload their DL task to the edge server via blockchain transactions.

Author Contributions: Conceptualization, S.R.; methodology, S.R.; software, S.R.; validation, S.R.; formal analysis, S.R.; investigation, S.R.; writing—original draft preparation, S.R..; writing—review and editing, S.R., J.H.P; supervision, J.H.P; project administration, J.H.P.; funding acquisition, Y.P., J.H.P.

Funding: This study was supported by the Advanced Research Project funded by the SeoulTech (Seoul National University of Science and Technology).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Rathore, S.; Park, J.H. Semi-supervised learning based distributed attack detection framework for IoT. *Appl. Soft Comput.* **2018**, *72*, 79–89. [CrossRef]
2. Rathore, S.; Sharma, P.K.; Sangaiah, A.K.; Park, J.J. A hesitant fuzzy based security approach for fog and mobile-edge computing. *IEEE Access* **2017**, *6*, 688–701. [CrossRef]
3. The Future of Data with the Rise of the IoT. Available online: https://www.rfidjournal.com/articles/view?17954 (accessed on 11 April 2019).
4. Mohammadi, M.; Al-Fuqaha, A.; Sorour, S.; Guizani, M. Deep learning for IoT big data and streaming analytics: A survey. *IEEE Commun. Surv. Tutor.* **2018**, *20*, 2923–2960. [CrossRef]
5. Park, J.H. Practical approaches based on deep learning and social computing. *J. Inf. Process. Syst.* **2018**, *14*, 1–5.
6. Zhang, Q.; Yang, L.T.; Chen, Z.; Li, P.; Deen, M.J. Privacy-preserving double-projection deep computation model with crowdsourcing on cloud for big data feature learning. *IEEE Internet Things J.* **2018**, *5*, 2896–2903. [CrossRef]
7. Abeshu, A.; Chilamkurti, N. Deep learning: The frontier for distributed attack detection in fog-to-things computing. *IEEE Commun. Mag.* **2018**, *56*, 169–175. [CrossRef]
8. Li, H.; Ota, K.; Dong, M. Learning IoT in edge: Deep learning for the internet of things with edge computing. *IEEE Netw.* **2018**, *32*, 96–101. [CrossRef]

9. Wang, Z. Deep learning-based intrusion detection with adversaries. *IEEE Access* **2018**, *6*, 38367–38384. [CrossRef]
10. Edge, M.L. Deep Learning on IoT Devices. Available online: https://conferences.oreilly.com/oscon/oscon-or-2018/public/schedule/detail/67199 (accessed on 11 April 2019).
11. Tang, J.; Sun, D.; Liu, S.; Gaudiot, J.L. Enabling deep learning on IoT devices. *Computer* **2017**, *50*, 92–96. [CrossRef]
12. Rathore, S.; Kwon, B.W.; Park, J.H. BlockSecIoTNet: Blockchain-based decentralized security architecture for IoT network. *J. Netw. Comput. Appl.* **2019**, *143*, 167–177. [CrossRef]
13. Castro, M.; Barbara, L. Practical Byzantine fault tolerance. *OSDI* **1999**, *99*, 173–186.
14. Kim, H.W.; Jeong, Y.S. Secure authentication-management human-centric scheme for trusting personal resource information on mobile cloud computing with blockchain. *Hum. Cent. Comput. Inf. Sci.* **2018**, *8*, 1–11. [CrossRef]
15. Johnsonm, R.; Zhang, T. Accelerating Stochastic Gradient Descent Using Predictive Variance Reduction. In Proceedings of the NIPS, Lake Tahoe, NV, USA, 5–10 December 2013; pp. 1–10.
16. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436–444. [CrossRef] [PubMed]
17. Rathore, S.; Ryu, J.H.; Sharma, P.K.; Park, J.H. DeepCachNet: A Proactive Caching Framework Based on Deep Learning in Cellular Networks. *IEEE Netw.* **2019**, *33*, 130–138. [CrossRef]
18. Sharma, P.K.; Rathore, S.; Park, J.H. DistArch-SCNet: Blockchain-based distributed architecture with li-fi communication for a scalable smart city network. *IEEE Consum. Electron. Mag.* **2018**, *7*, 55–64. [CrossRef]
19. Fouque, P.A.; Poupard, G.; Stern, J. Sharing Decryption in the Context of Voting or Lotteries. In *Proceedings of the International Conference on Financial Cryptography*; Springer: Berlin/Heidelberg, Germany, 2000; pp. 90–104.
20. Everingham, M.; Eslami, S.A.; Van Gool, L.; Williams, C.K.; Winn, J.; Zisserman, A. The pascal visual object classes challenge: A retrospective. *Int. J. Comput. Vis.* **2015**, *111*, 98–136. [CrossRef]
21. Hazay, C.; Mikkelsen, G.L.; Rabin, T.; Toft, T. Efficient RSA Key Generation and Threshold Paillier in the Two-Party Setting. In *Proceedings of the Cryptographers' Track at the RSA Conference*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 313–331.