Blockchain based End-to-end Protection of IoT Intelligence Application in Edge Computing Environment

Lei Xu, Zhimin Gao, Xinxin Fan, Lin Chen, Hanyee Kim, Taeweon Suh, and Weidong Shi

Abstract—**IoT devices provide a rich data source that is not available in the past and valuable for a wide range of intelligence applications such as deep neural networks. The analysis results that are generated by these intelligent applications are then sent back to IoT devices to improve the operation of the system. The progress in distributed deep neural network training enables the deployment of training modules to edge data centers that are close to the IoT devices to reduce the latency and movement of large amounts of data, which makes the integration of IoT and edge computing more attractive. However, these IoT devices and edge data centers are usually owned and managed by different parties. It is hard to coordinate them to provide an end-to-end protection of the intelligence applications with classical security enhancement tools. To mitigate this risk, we propose a novel blockchain based end-to-end IoT intelligence application protection system in the edge computing environment. The protection system leverages a set of cryptography primitives to build a blockchain adapted for edge computing that is scalable to handle a large number of IoT devices. The customized blockchain is integrated with a distributed deep neural network to offer integrity and authenticity protection service.**

Index Terms—**blockchain, IoT, DNN, security**

I. Introduction

Internet of things (IoT) has become an essential part of many IT infrastructures such as smart city, smart factory, and smart farming. One of the major functions of IoT devices is information collection, and the collected data is then used to drive various intelligence applications, especially data hungry deep neural networks. Fusion of edge computing [1] and distributed deep neural network (DNN) training technologies [2] further accelerates the adoption of IoT based DNN applications. Specifically, IoT devices generated data are collected and processed by different edge data centers that are physically close to the data source to save communication cost and reduce latency. The intermediate results are then sent to the cloud and are merged to obtain the final result, which is in the form of a trained model for DNN. The final model is then sent back to edge servers to be utilized by different sub-systems to form a closed circle.

The trained DNN models can be used for different purposes, such as decision making support and critical infrastructure operation assistance. The failure of preserving the integrity of any step in this process can lead to an error and cause serious consequences [3] Therefore, it is of utmost importance to develop a protection mechanism for such a scenario. Although traditional cryptography tools like digital signature are useful to protect the process, there are several limitations: (i) Multiple IoT systems are managed by different participants and tradition cryptography tools do not support coordination between these participants; and (ii) Existing cryptography tools can only guarantee the integrity and authenticity of a single data item, but cannot protect the complete process directly.

The blockchain technology offers a promising approach to mitigate these challenges. In a nutshell, a blockchain is a decentralized ledger that is managed by multiple participants collaboratively without relying on a centralized party. The concept was first introduced to build cryptocurrency systems without a central trusted party and then finds a variety of other applications in different areas including supply chain [4] and sharing economy [5]. This decentralization feature matches with the feature of typical IoT based distributed DNN applications well. However, it is not straightforward to implement a blockchain based protection mechanism due to the characters of IoT devices and the edge computing architecture, and the complexity of DNN applications. The IoT devices usually have limited computation and storage capacity and cannot afford expensive blockchain operations. At the same time, the number of IoT devices involved in a typical DNN application is large, and it is not easy to handle them with a blockchain. Furthermore, most existing blockchains do not consider the special network architecture of edge computing to utilize the high speed connection within an edge data center.

To fully utilize the desirable features of blockchain to offer an end-to-end protection for intelligent applications that are based on IoT and take advantage of the edge computing, we propose a novel blockchain architecture and corresponding key operations. Intelligent applications can be quite different, and we focus on the protection of distributed deep neural networks (DNNs) in this paper. It consists of multiple sub-blockchains and each of them runs in an edge computing data center and serves a group of IoT devices. These sub-blockchains are not independent and are connected through another blockchain deployed in the cloud in an efficient manner.

In summary, our contributions to this work include:

- We propose a blockchain framework that fits the edge

L. Xu is with University of Texas Rio Grande Valley
Z. Gao is with Auburn University at Montogmery
X. Fan is with IoTeX
L. Chen is with Texas Tech University
H. Kim and T. Suh are with Korea University
W. Shi is with University of Houston

computing architecture and IoT based distributed DNN applications;
- We develop the detailed design of integrating the proposed blockchain framework with a typical DNN application to offer end-to-end protection; and
- We implement a prototype of the blockchain based AI protection mechanism and evaluate its performance to demonstrate its practicality.

The rest of the paper is organized as follows: In Section II, we briefly review related background. In Section III, we describe the high level design of the two-layer edge blockchain architecture and its application in the protection of IoT based DNN applications. We present the detailed design and analysis of the new blockchain architecture and the protection of DNN in Section IV and Section V respectively. In Section VI, we discuss the implementation of the design and evaluate its performance. In Section VII, we review related prior works, and we conclude the paper in Section VIII.

II. Background

In this section, we briefly review the background of blockchain and cryptography accumulator, which is used for the construction of the new blockchain system.

A. Blockchain

There are generally two types of blockchain systems based on the way of participant management. One type is public blockchain, where there is no central identity/authorization management system and anyone can join the system freely. The other type is permissioned blockchain, where an entity needs to be enrolled and authorized to operate in the blockchain. Although the IoT devices of a system belong to different owners, it is still a closed system, and it is a natural requirement for all participants to know each other. Therefore, we only consider permissioned blockchain in this work.

The most common way to identify participants in a permissioned blockchain system is to use a public key infrastructure (PKI). Each participant peer is assigned a public/private key pair (pk, sk) and the public key pk is then embedded into a certificate issued by a CA of the PKI. The certificate serves as the identity of the peer and the peer can authenticate him/herself to others by generating digital signatures with corresponding private key. Peers of a permissioned blockchain will only run a consensus protocol on transactions that are correctly signed by a known peer.

B. Cryptography Accumulator

The new blockchain architecture developed in this paper and its application in IoT based DNN protection utilize the cryptography accumulator as a building block. The cryptography accumulator has many applications and we consider membership management in this paper. Informally, a cryptography accumulator consists of four algorithms:
- `Setup`. This algorithm takes as input the security parameter κ, and outputs a set of public parameters *para*.
- `Update`. This algorithm takes as input a new member needs to be added to the current accumulator and *para*. It outputs an updated accumulator.
- `ProofGen`. This algorithm takes as input the member in question, the current accumulator, and *para*. It outputs a proof if the member belongs to the current accumulator.
- `ProofVerf`. This algorithm takes as input the member in question, the proof, the current accumulator, and *para*. It outputs 1 if the member belongs to the accumulator, and outputs 0 otherwise.

Because of the every-growing feature of blockchain, we do not consider the revoking operation of the cryptography accumulator scheme. A concrete construction of cryptography accumulator is provided in Section IV.

III. Overview of the Protection System

In this section, we provide an overview of the protection system that utilizes a customized blockchain architecture, the edge blockchain, to enhance the security of the distributed DNN application that relies on an IoT system.

The edge blockchain works in the edge computing environment to serve a large number of IoT devices and the DNN applications built on top of it. As the edge blockchain and the applications that it protects are not open systems, i.e., all entities (IoT devices, servers supporting the application, and peers maintaining the ledgers) need to be authorized before they can join the system and communicate with others. Therefore, the edge blockchain works in a permissioned manner and each entity is equipped by a public/private key pair, where the public keys are certified by a PKI system.

Fig. 1 depicts the overall architecture of edge blockchain and the way it is integrated with an edge computing infrastructure. The edge blockchain system consists of a group of sub-blockchains and a main-blockchain. A sub-blockchain is maintained by a number of peers running in the same edge data center and mainly serves the set of IoT devices connected to the same edge data center. A main-blockchain is set up in the cloud to connect all sub-blockchains, and maintained by peers running in the cloud. Both sub-blockchains and main-blockchain peers are controlled by owners of IoT devices and other stakeholders. The main-blockchain does not connect to IoT devices directly, but works as a bridge to enable communications between sub-blockchains. Each sub-blokchain occasionally sends its current status to the main-blockchain and the main-blokchain also shares its status with all sub-blockchains. This inter-locking mechanism guarantees that even if an adversary takes over some of the sub/main-blockchains, the immutability feature is still preserved.

To protect a DNN application built on top of an IoT system, the edge blockchain is utilized to track every step of the lifecycle of the application. As demonstrated in Fig. 1, IoT devices generated data is collected and processed by the corresponding edge data center. The sub-blockchain running in the same edge data center keeps a record of the collected data in an immutability manner. The computation process is protected by treating intermediate results as data in the

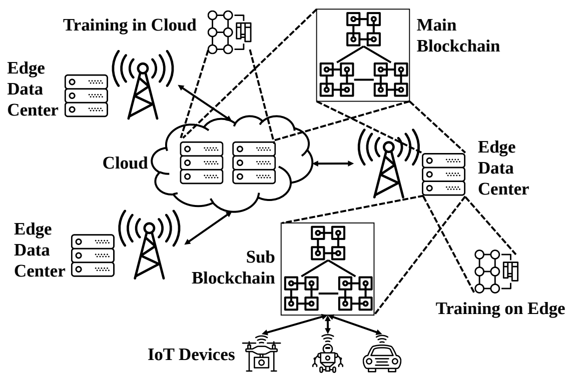

Fig. 1: Overview of the edge blockchain architecture. Each edge data center maintains its own sub-blockchain, which serves a set of IoT devices and connects to the main-blockchain system resides on the cloud data center. Occasionally, a sub-blockchain submits its status to the main-blockchain, and the main-blockchain shares its status with all sub-blockchains. Under the edge blockchain architecture, both IoT devices and computers used for model training are end users of the system.

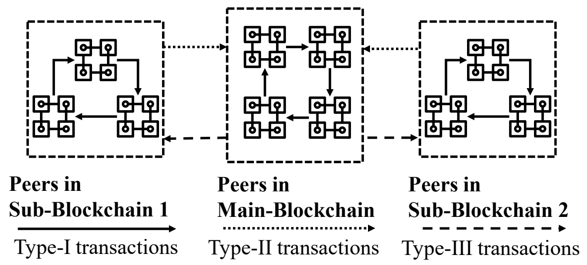

Fig. 2: Transaction processing in the edge blockchain architecture. Peers in an edge data center maintain a local sub-blockchain, and submit Type-II transactions to peers in the cloud. Cloud peers work together to build and maintain the main-blockchain with received transactions, and the main-blockchain sends Type-III transactions to all sub-blockchains periodically.

sub-blockchain. Similarly, the main-blockchain running in the cloud also keeps a record of the intermediate models generated in edge data centers, the model aggregation process, and the final result model to prevent them from being compromised. We provide more details on the IoT DNN application protection in Section V.

IV. Detailed Design of the Two-layer Edge Blockchain

In this section, we discuss the key design of the two-layer edge blockchain architecture. To simplify the description, we assume each block only contains one transaction. The system can be easily extended to handle the case where multiple transactions are packed into a single block.

A. Basic Inter-locking of Blockchains with RSA Accumulator

In order to inter-lock the sub-blockchains and the main-blockchain in the two-layer edge blockchain system and efficiently enable transaction validity verification, the RSA based accumulator can be utilized as described in [6]. There are three types of transactions in the edge blockchain system and their relationship is summarized in Fig. 2.

- Type-I: ordinary transactions within a sub-blockchain and the main-blockchain. This transaction type keeps information from data generated by IoT devices and machine learning algorithms running in edge/cloud data centers. Type-I transactions are the origination of other types of transactions.
- Type-II: transactions submitted to the main-blockchain by sub-blockchains. The main-blockchain is responsible for keeping records of the status of sub-blockchains, so each sub-blockchain needs to send their information to peers of the main-blockchain running in the cloud, which is derived from Type-I transactions.
- Type-III: transactions submitted to sub-blockchains by the main-blockchain. The purpose of this type of transactions is to disseminate the status of the main-blockchain and prevent an adversary from breaking the integrity of the system by compromising peers running in the cloud. Note that this type of transactions also include information collected from sub-blockchains.

Both Type-II and Type-III transactions are generated from Type-I transactions directly/indirectly. The reason to introduce these two new types of transactions to support inter-blockchain information exchange is that if all sub-blockchains send original transactions to the main-blockchain, and the main-blockchain forwards all received transactions to all sub-blockchains, it equivalents to the case that each blockchain (sub or main) stores all information of the whole system directly, which is too expensive and unnecessary. Therefore, an RSA accumulator is used to compress a blockchain to reduce the cost [7], [6]. The scheme consists of four algorithms:

- `Setup`. This algorithm is executed when a sub-blockchain or main-blockchain is initialized. The creator of the blockchain selects two large prime numbers p and q, the sizes of which are determined by the security parameter κ. The creator then calculates $N \leftarrow p \cdot q$ and and selects a random value $g \leftarrow \mathbb{Z}_N^*$. p, q are discarded and N, g are public parameters of this blockchain.
- `Update`. This algorithm is used by a blockchain peer to update the compressed value. For the genesis block, it calculates

$$v_1 \leftarrow g^{hash(blk_1||1)} \mod N,$$

where *hash* is a cryptographic hash function that is collision resistant, and *blk*$_1$ is the contents of the genesis block. The number 1 contacted to *blk*$_1$ is the sequence number of the block. For the ith block where $i > 1$, the algorithm calculates

$$v_i \leftarrow v_{i-1}^{hash(blk_i||i)} \mod N.$$

- `ProofGen`. This algorithm is used by a proving peer to generate a proof of a given block. In order to show a block blk' is the ith block of a blockchain with the accumulator value v and n blocks in total, the algorithm computes a proof (ρ_1, ρ_2) where $\rho_1 \leftarrow hash(blk'||i)$ and $\rho_2 \leftarrow g^{\prod_{k=1}^{n} hash(blk_k||k)/\rho_1} \mod N$.
- `ProofVerf`. This algorithm is used by a verifier to check the validity of a proof for block blk'_i of blockchain with accumulator value v_n. Assume the proof is (ρ'_1, ρ'_2), the algorithm checks $\rho'_1 \overset{?}{=} hash(bkl'_i||i)$ and $v_n \overset{?}{=} {\rho'_2}^{\rho'_1} \mod N$. If both equations hold, the block in question is valid. Otherwise the transaction is not valid.

When a new block is added to a specific blockchain, an updated accumulator is also attached to it. A Type-II transaction sending from a sub-blockchain to the main-blockchain is the most recent accumulator value of the sub-blockchain, and a Type-III transaction is the latest accumulator of the main-blockchain.

B. Cross Blockchain Verification

When there is only one blockchain, it is straightforward to utilize the cryptography accumulator for transaction verification, i.e., a party with the latest accumulator value can check whether a transaction is included in the blockchain. The situation is more complex for the two-layer edge blockchain architecture. Without loss of generality, we consider the scenario a transaction tx_Q initialized in sub-blockchain $blkc_B$ needs to be verified by a verifier peer p_{vrfer} in sub-blockchain $blkc_A$. Protocol 1 summarizes the transaction verification process of this case.

Protocol 1 The basic cross blockchain transaction verification protocol.

```
Input: The transaction $tx_Q$, and sub-blockchain information $blkc_A, blkc_B$.
Output: $b \leftarrow 1$ if $tx_Q$ is valid; $b \leftarrow 0$ otherwise.
 1: $p_{vrfer}$ obtains the latest accumulator $acc_M$ of the main-blockchain from the local copy of $blkc_A$;
 2: $p_{vrfer}$ interacts with main-blockchain to retrieve transaction $tx_1$, which represents the most recent accumulator of $blkc_B$, and according proof $pf_M$;
 3: $p_{vrfer}$ runs $b_1 \leftarrow \texttt{Verification}(tx_1, acc_M, pf_M)$;
 4: if $b_1 = \texttt{TRUE}$ then
 5:     $p_{vrfer}$ extracts the accumulator of $blkc_A$ as $acc_A \leftarrow \texttt{Extract}(tx_1)$;
 6:     $p_{vrfer}$ interacts with $blkc_A$ to obtain a proof $pf_A$ for $tx_Q$;
 7:     $p_{vrfer}$ runs $b_2 \leftarrow \texttt{Verification}(tx_Q, acc_A, pf_A)$;
 8:     if $b_2 = \texttt{TRUE}$ then
 9:         return $b \leftarrow 1$;
10:     else
11:         return $b \leftarrow 0$;
12:     end if
13: else
14:     return $b \leftarrow 0$;
15: end if
```

C. Secure System Initialization

The original design of the RSA accumulator [7] requires a secure setup, i.e., a trusted third party is needed to generate the modular value N and destroys the two corresponding large prime factors p, q. If the factorization is leaked to an attacker, he/she can generate a proof for an arbitrary transaction that can pass the verification. In the decentralized environment we consider, there is a lack of such trusted third party. We utilize a distributed RSA parameter generation scheme given in [8] to over come this limitation.

A set of k peers are selected to work together to generate the modulus N as follows:

- One server proposes a large prime number P that is larger than the target modulus value, and all peers reach agreement on P.
- For peer $i = 1, \dots, k$, it selects two numbers p_i, q_i, two random polynomials $f_i, g_i \in \mathbb{Z}_P[x]$ such that $f_i(0) = p_i, g_i(0) = q_i$, and a third random polynomial $h_i \in \mathbb{Z}_P[x]$ such that $h_i(0) = 0$.
- For peer $i = 1, \dots, k$, it computes $p_{i,j} = f_i(j), q_{i,j} = g_i(j), h_{i,j} = h_i(j)$ for $j = 1, 2, \dots, k$. Peer i privately sends $(p_{i,j}, q_{i,j}, h_{i,j})$ to peer j that $i \neq j$.
- For peer $i = 1, \dots, k$, it computes
$$N_i = (\sum_{j=1}^{k} p_{j,i})(\sum_{j=1}^{k} q_{j,i}) + \sum_{j=1}^{k} h_{j,i} \mod P,$$
and broadcasts N_i to other peers.
- For peer j, it evaluates polynomial $\alpha(x) = (\sum_j f_j(x))(\sum_j g_j(x)) + \sum_j h_j(x) \mod P$ at 0, and we have $\alpha(0) = N$.
- All peer run a distributed primality test to check whether N is a product of two prime numbers. If N does not pass the test, the process is repeated.

We refer the readers to [8] for more details on the algorithm and its complexity. Note that for our scenario, we only need the modulus value N but do not need to generate a public/private key pair in a distributed manner. The generated value N can be shared with all sub-blockchains and the main-blockchain, so the generation process does not need to be repeated by each individual blockchain. Furthermore, N only needs to be generated once so the performance of this process is not critical for the system. Each sub-blockchain and the main-blockchain can select its own generator for the accumulator, which can be done by any peer and does not affect the security.

D. Efficient Batch Proof Generation

The IoT based DNN application may generate a large number of requests on information verification. Therefore, it is helpful to generate multiple proofs together to reduce the computation cost compared with generating these proofs individually. For a specific sub/main-blockchain, we assume it has

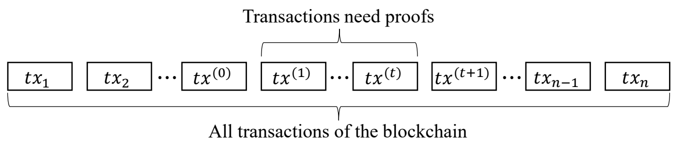

Fig. 3: Efficient batch proof generation. Transactions before and after the target set of transactions only need to be processed once to save computation cost.

totally n transactions, and needs to generate $t < n$ proofs for transactions $tx^{(1)}, tx^{(2)}, \ldots, tx^{(t)}$, which have been ordered by their appearance order in the blockchain. Fig. 3 demonstrates the setting, and the batch proof generation process is given in Protocol 2.

Protocol 2 Batch proof generation protocol.

```
Input: $blkc, tx^{(1)}, \ldots tx^{(t)}$.
Output: Proofs $g^{(1)}, g^{(2)}, \ldots, g^{(t)}$.
1: Obtain the accumulator $g_0$ attached to $tx^{(0)}$;
2: for transactions $tx^{(t+1)}$ to $tx_n$ do
3:     $g_0 \leftarrow \texttt{Update}(g_0, tx^{(j)})$;
4: end for
5: for $i = 1$ to $t$ do
6:     Update $g_0$ accordingly to obtain $g^{(i)}$;
7: end for
8: return $g^{(1)}, g^{(2)}, \ldots, g^{(t)}$;
```

E. Efficient Batch Proof Verification

To verify a proof of a transaction, the verifier computes and compares:

$$\begin{aligned}(g^{x_1 x_2 \ldots x_{i-1} x_{i+1} \ldots x_n})^{x_{i_1}} \mod N &\overset{?}{=} X_1 \Leftrightarrow \\ g_1^{x_{i_1}} \mod N &\overset{?}{=} X_1\end{aligned}$$

where $g^{x_1 x_2 \ldots x_{i-1} x_{i+1} \ldots x_n} = g_1$, X_1 is the accumulator value, and $x_1, x_2, \ldots, x_n$ are values derived from corresponding blockchain transactions. When there are t instances need to be verified, it is equivalent to the evaluation and comparison of the following:

$$\begin{cases} g_1^{x_{i_1}} \mod N & \overset{?}{=} X_1 \\ g_2^{x_{i_2}} \mod N & \overset{?}{=} X_2 \\ & \cdots \\ g_t^{x_{i_t}} \mod N & \overset{?}{=} X_t \end{cases}$$

Instead of evaluating these modular exponentiation one by one, we propose a more efficient approach to evaluate them together by computing

$$g_1^{x_{i_1}} \cdot g_2^{x_{i_2}} \cdot \ldots \cdot g_t^{x_{i_t}} \overset{?}{\equiv} X_1 \cdot X_2 \cdot \ldots \cdot X_t \mod N \quad (1)$$

We first describe the verification algorithm, and then analyze its performance and demonstrate the security of the aggregated proof verification.

Computation of batch proof verification. To simplify the description, we assume the sizes of the processed transactions (i.e., $x_{i_1}, x_{i_2}, \ldots, x_{i_t}$) have the same size. In practice, this can be done by adding leading 0s to those with less number of bits, and initialize the value on the left side of Equation (1) accordingly. The detailed batch proof verification process is given in Protocol 3.

Protocol 3 Efficient batch proof verification.

```
Input: Proofs $g_1, g_2, \ldots, g_t$, transactions $x_{i_1}, x_{i_2}, \ldots, x_{i_t}$, accumulators $X_1, X_2, \ldots, X_t$, and the public parameter $N$.
Output: $b = 1$ if all transactions are valid; $b = 0$ otherwise.
1: $g \leftarrow g_1 \cdot g_2 \cdot \ldots \cdot g_t \mod N$;
2: for $j \leftarrow |x_{i_1}|_b - 1, j \geq 0$ do
3:     $g \leftarrow g^2 \mod N$;
4:     Read the $j$th bit of $x_{i_1}, \ldots, x_{i_t}$ to form a bit vector $v_j$;
5:     $g_{v_j} \leftarrow \prod_{v_j[m]=1} x_{i_m} \mod N$;
6:     $g \leftarrow g \cdot g_{v_j} \mod N$
7:     $j \leftarrow j - 1$;
8: end for
9: $X \leftarrow X_1 \cdot X_2 \cdot \ldots \cdot X_t \mod N$;
10: if $g = X$ then
11:     return $b \leftarrow 1$;
12: else
13:     return $b \leftarrow 0$;
14: end if
```

Performance analysis. The idea of the batch verification method given in Protocol 3 is to aggregate the verification of multiple exponent computations into a single one, and a key step is the update operation given by Line 5 of Protocol 3. Instead of doing the computation on the fly every time, we do this by pre-computation. For t verifications, 2^t values are generated and stored to facilitate the verification, which represents products of all different combinations of the proofs. The pre-computation is not feasible for a large value of t, and we compare the computation cost for $t = 8$ and $|N|_b = 2048$.

- Independent verification. When a square-multiplication algorithm is used to evaluate the modular exponentiation for verification, one verification requires 2,048 multiplications/squares. The overall computation cost is 16,384 multiplications/squares.
- Batch verification. The pre-computation requires $2^8 = 256$ multiplications. The square-multiplication algorithm only needs to be executed once to finish the verification, which costs 2,048 multiplications/squares. The total computation cost is 2,034 multiplications/squares.

In summary, under this configuration, the batch verification computation cost is only about 12.4% of the independent verification. In practice, the computation saving can be less as the square operation is cheaper than general multiplication.

Security of batch proof verification. The accumulators are periodically backed up to different blockchains and we assume $X_1, X_2, \ldots, X_t$ are always correct. When the batch verification method given in Protocol 3 is applied and the attacker wants to cheat the verifier with a transaction that does not exist on the blockchain, he/she needs to produce at least two transactions with corresponding proofs. Without loss of generality, we consider the case where the attacker targets at accumulators X_1 and X_2. The attacker needs to find out g_1', g_2' and x_{i_1}', x_{i_2}' such that

$$g_1'^{x_{i_1}'} \cdot g_2'^{x_{i_2}'} = X_1 \cdot X_2 \mod N,$$

where $x_{i_1}' \neq x_{i_1}$ and $x_{i_2}' \neq x_{i_2}$. Note that the attacker also has the freedom to choose the proofs g_1', g_2', and it is possible that the attack manages to find out such two pairs. However, finding x_{i_1}' and x_{i_2}' is not enough as the attacker's goal is to convince the verifier that a transaction(s) is included in a blockchain, and a conversion algorithm is applied to the transaction to obtain corresponding value used in the verification. If we assume the hash function works as an oracle, even if the attacker can find out the values used in the verification, he/she cannot generate corresponding fake transactions.

V. Detailed Design of the Protection of IoT based DNN Applications with Edge Blockchain

Although there are variety types of DNNs that can be built on an IoT system in the edge computing environment, most of them can be roughly divided into three components: (i) Data collection. IoT devices are connected to edge data centers that are close to them and send/receive data to/from these edge data centers. (ii) Model construction. Given a neural network structure and a set of unknown parameters, the training process uses collected data to determine the values of the parameters. Since data are collected and consumed by different edge data centers to reduce communication cost, the training is done in a distributed manner, i.e., a local model is trained within each edge data center using data collected by connected IoT devices and the aggregated model is constructed using all local models in the connected cloud. The aggregated model may be sent back to all edge data centers to further improve the local models. (iii) Model application. Applying a completed model is straightforward, i.e., the model takes as input the new data and outputs the result, which can be a prediction or classification result depending on the nature of the model. The model application can be done either on edge data centers or cloud.

To integrate the two-layer edge blockchain with an IoT based DNN application and offer an end-to-end protection, the system needs to satisfy three requirements:

- The data collection and training process are recorded as blockchain transactions, which are distributed to all peers in the system.
- When the edge blokchain as a whole is secure, an adversary cannot compromise the integrity of the processes, e.g., the adversary cannot alter collected data, modify parameters of a trained model, or cheat in the model application.
- A third party can verify the integrity/authenticity of the whole lifecycle of an IoT based DNN application efficiently with information stored in edge blockchain.

A. Protection of Data Collection

When edge computing is utilized to facilitate an IoT based DNN application, IoT devices connect to different edge data centers based on their physical locations and submit collected data for model construction, as depicted in Fig. 1. Data collected by these IoT devices are used by local model training algorithm and the integrity and authenticity of these data are the prerequisites for the integrity of the model training procedure.

For an IoT device d^E connecting to a peer p^E in the edge data center E, it is served by the sub-blockchain *blkc*^{E}, which is maintained by peers running in E. When the device d^E collects new data d, it is processed as follows:

- Integrity/authenticity tag generation. Since we assume each IoT device is equipped with a public/private key pair, d^E signs d with its private key and the digital signature σ is used to protect the integrity and authenticity of the data. Note that in case the IoT device has limited computation power or energy supply, d^E can delegate signature generation to the connected peer p^E.
- Transaction construction and storage. To reduce the storage cost of the sub-blockchain, collected data d is saved to a separate storage system in the edge data center. A Type-I transaction is formed as (d^E, σ, prt) where *prt* is a pointer to d, which is saved to the sub-blockchain *blkc*^{E}.
- Transaction dissemination. A set of transactions in the form of (d^E, σ, prt) is then converted to a Type-II transaction (i.e., an updated accumulator), which is then submitted to main-blockchain *blkc*^{M} for storage.

A third party can connect with any peer of the edge blockchain system and interact with it to check whether a collected data d is tampered using the cross blockchain transacting verification method given in Protocol 1.

B. Protection of Model Training Process

Distributed training of DNN models have received extensive attention [9], [10], [11], which fits the edge computing environment very well, i.e., IoT devices send data to connected edge data centers to build sub-models and the sub-models are merged to obtain the final model in the cloud.

Since the final model depends on multiple local models trained in edge data centers, we first describe the protection of the training of a local model with a sub-blockchain that resides on the same edge data center. The idea is converting the model training protection to data protection. A local model is trained through multiple rounds, and parameters of the model are improved in an incremental way. For each round, the training algorithm takes as input some of the IoT devices collected data and the intermediate model generated in the previous round, and outputs a new intermediate model. More formally, let $(d_1, d_2, \ldots)$ denote the sequence of data sets collected by local IoT devices for training, $\mathcal{M}_i$ denote the current local model generated by the training algorithm $\mathcal{T}$, and $\mathcal{M}_O$ denotes the aggregated model from the cloud, we have

$$\mathcal{M}_{i+1} \leftarrow \mathcal{T}(\mathcal{M}_i, \mathcal{M}_O, d_{i+1}), i = 1, 2, \ldots, \tag{2}$$

where $\mathcal{M}_0$ is the initial local model with pre-defined parameters.

According to the calculation process given in Equation (2), it is easy to see that the integrity of $\mathcal{M}_{i+1}$ depends on the integrity of $\mathcal{M}_i, \mathcal{M}_O$, and d_{i+1}. Two functions are needed to protect the model training process: (i) Storage of model training process. This function utilizes the consensus and immutability features of the blockchain to solidify the training process. (ii) Verification of stored model training process. This function allows a third party to verify the integrity of the training process in an efficient manner.

Storing model training process with the sub-blokchain. The model training process is stored with the sub-blockchain by saving intermediate results, which are in fact intermediate models when we consider DNN. The protocol of storing a single intermediate model $\mathcal{M}_{i+1}$ works as follows:

- Preparing and converting an intermediate training result into a blockchain transaction:
  - Before calculating a new intermediate result $\mathcal{M}_{i+1}$ with algorithm $\mathcal{T}$, one checks the integrity of $\mathcal{M}_O$ with the main-chain through cross blockchain transaction verification. $\mathcal{M}_i$ and d_{i+1} are local and their integrity is easy to verify with the sub-blockchain;
  - A peer runs $\mathcal{T}(\mathcal{M}_i, \mathcal{M}_O, d_{i+1})$ to get $\mathcal{M}_{i+1}$. Note that the inputs and outputs are stored locally in the edge computing data center.
  - A peer converts the intermediate training result into a transaction by computing $tx_{\mathcal{M}_{i+1}} \leftarrow hash(\mathcal{M}_{i+1}||\mathcal{M}_O||d_{i+1}||i+1)$.
- Storing the transaction of an intermediate training result to corresponding sub-blockchain:
  - A peer submits $tx_{\mathcal{M}_{i+1}}$ to the sub-blockchain in the same edge computing data center;
  - Each peer of the sub-blockchain verifies intermediate result $\mathcal{M}_{i+1}$ to determine whether to accept $tx_{\mathcal{M}_{i+1}}$ or not;
  - A consensus protocol is executed between the sub-blockchain peers. Depending on the consensus protocol, if a certain percentage/number of peers agree on the result $\mathcal{M}_{i+1}$, $tx_{\mathcal{M}_{i+1}}$ is accepted in the sub-blockchain.

The inter-locking mechanism described in Section IV is applied to disseminate information of the transaction to the whole system.

Verification of the model training process. The purpose of storing model training process in the edge blockchain system is to allow a third party to verify the integrity of the training process without repeating the expensive training process. Without loss of generality, we consider the verification of a local model $\mathcal{M}$. Note that $\mathcal{M}$ is generated in multiple steps using a sequence of training data sets, and we assume the number of intermediate models is ℓ. The verifier repeat following steps until satisfied:

- The verifier randomly selects a number $i \in [1, \ell]$;
- The verifier requires $\mathcal{M}_i, \mathcal{M}_{i-1}, d_i$ and the external model information $\mathcal{M}_O$ from the edge data center;
- The verifier checks the integrity of received data with the cross blockchain transaction verification mechanism given in Protocol 1. If the data is not consistent with information stored in edge blockchain system, the verifier rejects $\mathcal{M}$;
- The verifier re-compute $\mathcal{M}_i' \leftarrow \mathcal{T}(\mathcal{M}_i, \mathcal{M}_O, d_{i+1})$. If $\mathcal{M}_i' \neq \mathcal{M}_i$, the verifier rejects $\mathcal{M}_i$.

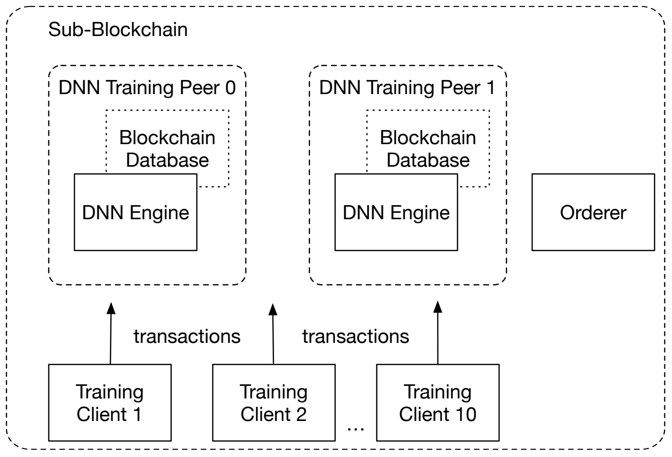

Fig. 4: Overview of the sub-blockchain prototype. During benchmarking, all transactions are driven via a training client, which is implemented by a Hyperledger Fabric gateway. A training client collects data from IoT devices and submits it to the training peers. Peers then train the local model with the data, convert the result to a hash, and store it in the blockchain. In this experiment, two training peers are involved in a sub-blockchain.

VI. Implementation and Experiments

In this section, we discuss the implementation of the proposed protection mechanism and provide preliminary performance evaluation.

In this experiment, both the cloud and the edge data centers utilize Hyperledger Fabric [12] as their basic permissioned blockchain systems. Hyperledger Fabric is an opensource system for deploying and operating permission-based blockchain hosted by Linux Foundation. For simplicity, every sub-blockchain is configured to have only one single channel. Transactions are deployed with a 2-of-any endorsement policy. In this way, we can eliminate the delay from the orderer because a single peer in this channel will not interact with the orderer. Each transaction inserts a single hash of the intermediate training result into the world state database. Fig. 4 illustrates the overall design of the sub-blockchain prototype.

Transaction creation. We conduct a simulation of 100 transactions for each of the 10 DNN training clients on a sub-blockchain (total 1000 transactions). Latency and throughput are investigated. The simulation is run on an AWS EC2 t2.2xlarge instance, which has 8 vCPUs, 32 GB memory and 30 GB SSD vDisk. TABLE I shows the evaluation results.

TX size (bytes)	Max Latency (s)	Avg. Latency (s)	Throughput
32 (SHA-256)	0.65	0.33	193.33
8K	0.70	0.41	173.04
16K	0.97	0.54	131.98
32K	2.35	1.09	65.69

TABLE I: Performance Evaluation for Transaction Creation

We assume that the transaction size is close to the hash size, which is generated from the intermediate training results. The latency consists of regular block creation time and accumulator calculation time. Note that the latency of proof generation is not included because it is an off-chain operation. We observe that both latency and throughput are affected significantly only when the size of a transaction becomes greater than 8K. In all cases, the latency caused by accumulator calculation is a constant number of 0.1 seconds approximately. As a result, transaction size is the major factor that determines the performance. However, it can be maintained in a small number by utilizing an appropriate hash function.

Transaction propagation. In the proposed system, Type-II and Type-III transactions are inter-chain transactions. Each sub-blockchain occasionally exchanges the current accumulators with main-blockchain. The performance of accumulator propagation is mainly affected by network delay, which will not be discussed here.

VII. Related Works

In this section, we briefly review related works.

Leveraging blockchain to protect IoT systems and their applications has received extensive attention. There are a large number works that utilize blockchain as a black box to protect IoT systems and applications built on top of them [13], [14], [15]. Biswas *et al.* proposed a blockchain scheme that is customized for IoT data protection [16], which also adopted a two-layer structure to improve scalability to handle a large number of IoT devices. Their approach simply restricted the number of transactions that can be sent to the global blockchain and did not provide an efficient mechanism to allow the two layers to exchange information. A space-structured blockchain structure and a collaborative proof-of-work consensus protocol were developed to support an IoT system in [17]. This work aims at improving the performance of a single blockchain, which can be incorporated into the proposed two-layer blockchains framework. Xu et al. proposed a blockchain based IoT data protection mechanism which utilizes a similar way to organize multiple blockchains to handle a large number of IoT devices [18]. But they did not consider the integration with distributed DDN applications of IoT to offer protection, which can involve a large number of transaction validity proof generation/verification.

VIII. Conclusion

Integrating IoT and edge computing for new intelligence algorithms has many promising application scenarios. Some of these scenarios have a high security requirement because they involve critical infrastructures, and a failure to preserve an end-to-end integrity protection can have serious consequences. As a decentralized data structure with several desirable security features, it is a natural idea to apply blockchain to protect this type of systems. In this paper, we propose a two-layer edge blockchain system and corresponding operation performance improvement technologies that fit the characters of the edge computing environment. We also describe the way to integrate the novel blockchain system with the target DNN application that needs to be protected, and develop a prototype to demonstrate its performance and practicability.

References

[1] J. Pan and J. McElhannon, "Future edge cloud and edge computing for internet of things applications," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 439–449, 2017.

[2] T. Park and W. Saad, "Distributed learning for low latency machine type communication in a massive internet of things," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 5562–5576, 2019.

[3] "Artificial intelligence and cybersecurity: Opportunitiesand challenges."

[4] Z. Gao, L. Xu, L. Chen, X. Zhao, Y. Lu, and W. Shi, "Coc: A unified distributed ledger based supply chain management system," *Journal of Computer Science and Technology*, vol. 33, no. 2, pp. 237–248, 2018.

[5] L. Xu, N. Shah, L. Chen, N. Diallo, Z. Gao, Y. Lu, and W. Shi, "Enabling the sharing economy: Privacy respecting contract based on public blockchain," in *Proceedings of the ACM Workshop on Blockchain, Cryptocurrencies and Contracts*. ACM, 2017, pp. 15–21.

[6] L. Xu, L. Chen, Z. Gao, S. Xu, and W. Shi, "Efficient public blockchain client for lightweight users," *EAI Endorsed Transactions on Security and Safety*, vol. 4, no. 13, 1 2018.

[7] J. Benaloh and M. De Mare, "One-way accumulators: A decentralized alternative to digital signatures," in *Workshop on the Theory and Application of of Cryptographic Techniques*. Springer, 1993, pp. 274–285.

[8] M. Malkin, T. D. Wu, and D. Boneh, "Experimenting with shared generation of rsa keys." in *NDSS*, 1999.

[9] J. Chen, X. Pan, R. Monga, S. Bengio, and R. Jozefowicz, "Revisiting distributed synchronous sgd," *arXiv preprint arXiv:1604.00981*, 2016.

[10] Y. Lin, S. Han, H. Mao, Y. Wang, and W. J. Dally, "Deep gradient compression: Reducing the communication bandwidth for distributed training," *arXiv preprint arXiv:1712.01887*, 2017.

[11] Z. Zhang, L. Yin, Y. Peng, and D. Li, "A quick survey on large scale distributed deep learning systems," in *2018 IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS)*. IEEE, 2018, pp. 1052–1056.

[12] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich *et al.*, "Hyperledger fabric: a distributed operating system for permissioned blockchains," in *Proceedings of the Thirteenth EuroSys Conference*, 2018, pp. 1–15.

[13] J. Qiu, D. Grace, G. Ding, J. Yao, and Q. Wu, "Blockchain-based secure spectrum trading for unmanned-aerial-vehicle-assisted cellular networks: An operator's perspective," *IEEE Internet of Things Journal*, vol. 7, no. 1, pp. 451–466, 2019.

[14] C. Lin, D. He, N. Kumar, X. Huang, P. Vijaykumar, and K.-K. R. Choo, "Homechain: A blockchain-based secure mutual authentication system for smart homes," *IEEE Internet of Things Journal*, 2019.

[15] S. He, W. Ren, T. Zhu, and K.-K. R. Choo, "Bosmos: A blockchain-based status monitoring system for defending against unauthorized software updating in industrial internet of things," *IEEE Internet of Things Journal*, 2019.

[16] S. Biswas, K. Sharif, F. Li, B. Nour, and Y. Wang, "A scalable blockchain framework for secure transactions in iot," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4650–4659, 2018.

[17] Y. Liu, K. Wang, K. Qian, M. Du, and S. Guo, "Tornado: Enabling blockchain in heterogeneous internet of things through a space-structured approach," *IEEE Internet of Things Journal*, 2019.

[18] L. Xu, L. Chen, Z. Gao, X. Fan, T. Suh, and W. Shi, "DIoTA: Decentralized-ledger-based framework for data authenticity protection in iot systems," *IEEE Network*, vol. 34, no. 1, pp. 38–46, 2020.