Blockchain based End-to-end Tracking System for Distributed IoT Intelligence Application Security Enhancement

Lei Xu
The University of Texas Rio Grande Valley, lei.xu@utrgv.edu

Zhimin Gao

Xinxin Fan

Lin Chen

Hanyee Kim

See next page for additional authors

Recommended Citation

Xu, L., Gao, Z., Fan, X., Chen, L., Kim, H., Suh, T., & Shi, W. (2020). Blockchain based End-to-end Tracking System for Distributed IoT Intelligence Application Security Enhancement. 19th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (IEEE TrustCom 2020).

Blockchain based End-to-end Tracking System for Distributed IoT Intelligence Application Security Enhancement

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, which is valuable for a wide range of intelligence applications, especially deep neural network (DNN) applications that are data-thirsty. An established DNN model in turn provides useful analysis results that can improve the operation of IoT systems. The progress in distributed/federated DNN training further unleashes the potential of integration of IoT and intelligence applications. When a large number of IoT devices deployed in different physical locations, distributed training allows training modules to be deployed to multiple edge data centers that are close to the IoT devices to reduce the latency and movement of large amounts of data. In practice, these IoT devices and edge data centers are usually owned and managed by different parties, who do not fully trust each other or have conflicting interests. It is hard to coordinate them to provide an end-to-end integrity protection of the DNN construction and application with classical security enhancement tools. For example, one party may share an incomplete data set with others, or contribute a modified sub DNN model to manipulate the aggregated model and affect the decision-making process. To mitigate this risk, we propose a novel blockchain based end-to-end integrity protection scheme for DNN applications integrated with an IoT 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/federated DNN to offer integrity and authenticity protection services.**

Index Terms—**Blockchain, IoT, DNN, Security**

I. Introduction

Internet of things (IoT) has become an essential part of a variety of IT infrastructures such as smart city, smart factory, and smart farming. One of the most important functions of IoT devices is information collection, which provides a rich data source that is not available in the past and valuable for data-driven intelligence applications like deep neural networks (DNNs). A trained DNN model in turn can be applied for different purposes, such as security enhancement [1], [2], predictive maintenance [3], [4], and healthcare applications [5], [6]. While there are some efforts on finishing the machine learning tasks on IoT devices themselves [7], it is more common to shift these computation-intensive jobs to computers.

For many scenarios, IoT devices are deployed in multiple physical locations.It is expensive to move all collected data to the same place in a timely manner and process them together to build a DNN model, especially for online learning applications [8]. The fusion of edge computing [9] and distributed/federated deep neural network (DNN) learning technologies [10], [11] offers an adequate framework to handle this situation. Specifically, IoT devices generated data are sent to edge data centers that are physically close to the data source, where data is consumed to produce intermediate results. The intermediate results are then sent to the cloud, where they are aggregated to obtain the final DNN model. The final model can be pushed back to edge servers to facilitate the operation of connected IoT devices, or repeat the above process by working on the newly received model with new IoT data and sending the updated model to the cloud again.

While distributed/federated DNN learning has many benefits, it also faces new security challenges. A type of realistic threats that has attracted attention is that an attacker can manipulate its contribution to the learning process to affect the aggregated model, such as model poisoning [12], [13] and data poisoning [14], [15], [16]. These attacks may cause serious consequences when the compromised model is used for critical tasks.

To mitigate the risk of data poisoning, methods are developed to detect malicious data by measuring its negative impacts on generated models or using new loss functions [17], [18], [19]. A similar idea is extended to detect model poisoning attacks, e.g., removing local models that have large negative impacts on the error rate of the global model or using other outlier detection mechanism to filter out certain local models [12], [20]. These methods can only detect the attacks in a probability manner and it is helpful to keep track of the whole training/application process to offer the retrospect capability.

An IoT system integrated with a distributed/federated DNN usually consists of multiple sub-systems that owned by different parties. These parties may not trust each other and even have conflict of interests. Therefore, it is not easy to use traditional security tools to build the tracking system to offer the following features: (i) Consistency of tracking history. While one party can use tools like digital signature to authenticate the data and intermediate results produced by IoT devices/computers managed by him/her, it is possible that an attacker sends different versions to other parties to cause confusions. (ii) Completeness of tracking history. Besides tracking what information has been contributed, it is better for the system to guarantee that the recorded information is the only information that has been produced and shared. (iii) Immutability of tracking history. The system should make it hard for an attacker to manipulate existing records that have been stored in the tracking system.

The blockchain technology sheds light on overcoming 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 [21] and then finds a variety of other applications in different areas including supply chain management [22] and supporting of sharing economy [23]. This decentralization feature matches with the feature of typical IoT based distributed/federated 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 a blockchain to offer an end-to-end tracking of an IoT based distributed/federated DNN application that is based on IoT and take advantage of the edge computing, we propose edge blockchain, a novel blockchain architecture and corresponding key operations. The edge blockchain consists of multiple *sub-blockchain*s and a *main-blockchain*. Each sub-blockchain runs in an edge computing data center and serves a group of IoT devices to track collected data and intermediate results. All computer nodes maintaining the same sub-blockchain reside in the same edge data center so messages can be exchanged efficiently and cheaply. The main-blockchain runs in the cloud and is responsible for connecting all sub-blockchains to prevent an attacker to focus on a single sub-blockchain to compromise the subset of the training tracking information. An accumulator based efficient information exchange protocol is developed to support the inter-locking of sub-blockchains and the main-blockchain, and query of training tracking records. This protocol greatly reduces the inter-chain communication cost (i.e., sub-blockchain to sub-blockchain and sub-blockchain to main-blockchain). The proposed edge blockchain architecture for training/application process tracking achieves a good balance between scalability and immutability, i.e., more sub-blockchains can be easily added to the system to support more IoT devices and the inter-locking of sub-blockchains/main-blockchain prevent an attacker from modifying tracking information on a sub-blockchain without compromising the whole system. Besides maintaining an end-to-end tracking, the new blockchain architecture can also be utilized to support other IoT intelligence applications protection mechanisms that are based on blockchain technology.

In summary, our contributions in this paper include:

- We propose a novel blockchain architecture, edge blockchain, that fits the edge computing environment and is able to support end-to-end tracking of IoT based distributed/federated DNN applications;
- We develop an efficient inter-blockchain locking and query mechanisms to improve the security and performance of the proposed blockchain system; and
- We implement a prototype of the proposed blockchain and integrate it with a typical DNN training process to evaluate its performance and 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. Distributed/Federated DNN Construction

As the name suggested, a distributed/federated DNN construction involves multiple parties who collaborate to train the model with their own data sets. Instead of sharing data directly with a centralized party, they build their own local models and share the model with a centralized party to aggregate them to form the final model [24], [25]. Distributed/federated DNN construction has several benefits compared with centralized DNN model training: (i) Improving the performance. Distributed/federated model training not only allows the work to be done in parallel, but also reduces the communication cost as collected data is consumed locally. (ii) Improving data privacy. Participants of distributed/federated DNN learning do not share data directly but only contribute local models trained by the data. Therefore, it provides better data privacy in certain cases.

The distribution feature also introduces a new security risk, where a participant can either inject compromised data to the local model construction process or poison the local model directly.

B. 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.

A 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.

C. *Cryptography Accumulator*

Cryptography accumulator can be used for membership verification and is an important building block for the proposed design. 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 ever-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 IoT based Distributed/Federated DNN Construction Tracking with Edge Blockchain

In this section, we provide an overview of the IoT based distributed/federated DNN construction tracking mechanism that utilizes the edge blockchain architecture.

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. All entities of the system, including IoT devices, servers supporting the application, and peers maintaining the edge blockchain are authorized and obtain their identities before they can join the system and communicate with others. An identity is in the form of a public/private key pair where the public key is certified by a PKI system.

Fig. 1 depicts the high-level architecture of edge blockchain and the way it is integrated with an edge computing infrastructure to support distributed/federated DNN construction tracking. The edge blockchain system comprises 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 to track data collected by the set of IoT devices connected to the edge data center and local model. A main-blockchain is set up in the cloud to connect all sub-blockchains, and maintained by peers running in the cloud. The main-blockchain does not connect to IoT devices directly, but works as a bridge to enable communications between sub-blockchains, and is responsible for tracking collected local models and the model aggregation process. To integrate information stored in different components, each sub-blockchain occasionally sends its current status to the main-blockchain and the main-blokchain also broadcasts its status to all sub-blockchains. This inter-locking mechanism guarantees that even if an attacker takes over some of the sub/main-blockchains, the immutability feature is still preserved.

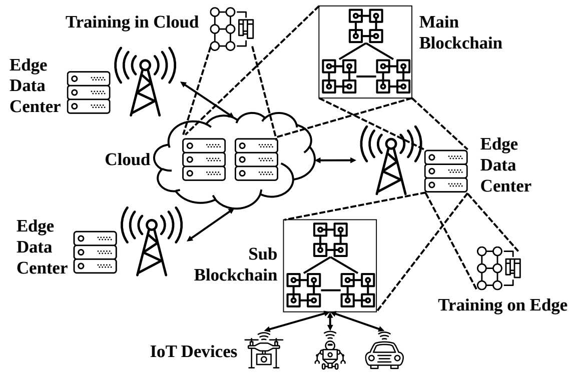

Fig. 1: Overview of the edge blockchain architecture. Each edge data center maintains its own sub-blockchain, which serves a set of IoT devices and connects to the main-blockchain system resides in the cloud data center. Occasionally, a sub-blockchain submits its status to the main-blockchain, and the 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.

To protect the distributed/federated 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 and generated local model in an immutability manner. 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.

IV. Detailed Design of the Edge Blockchain

In this section, we discuss the key design of the edge blockchain architecture. To simplify the description, we assume each block only contains a single 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*

To inter-lock the sub-blockchains and the main-blockchain in the edge blockchain system and enable efficient verification of transaction validity, the RSA based accumulator is utilized, which was used to support lightweight blockchain client [26]. There are three types of transactions in the edge blockchain system and their relationship is summarized in Fig. 2.

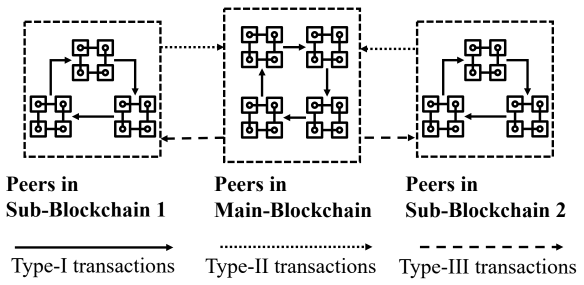

Fig. 2: Transaction processing in the edge blockchain architecture. Peers in an edge data center maintain a local sub-blockchain, and submit Type-II transactions to peers in the cloud. Cloud peers work together to build and maintain the main-blockchain with received transactions, and the main-blockchain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

- Type-I: ordinary transactions within a sub-blockchain or the main-blockchain. This type of transactions keep the tracking information, including data collected by IoT devices and intermediate results of the distributed/federated DNN construction. 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 collects statuses of sub-blockchains by asking each sub-blockchain to send their current status periodically to the main-blockchain peers running in the cloud. The status of the sub-blockchain is determined by all its existing blocks.
- 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 to sub-blockchains. The main-blockchain status is also determined by all its blocks. Note that these blocks include information of all connected sub-blockchains.

Type-I transactions are used to track local information, e.g., data collected by connected IoT devices and constructed local model (for a sub-blockchain), and aggregated model (for the main-blockchain). The goal of Type-II and Type-III transactions is to inter-lock sub-blockchains and the main blockchain, so even if an attacker takes over a sub-blockchain/the main-blockchain, he/she cannot modify the recorded tracking information without being detected. These transactions also need to allow a peer to verify whether a transaction is included in another sub-blockchain/the main-blockchain that it does not belong to in an efficient manner.

The key challenge here is how to build Type-II and Type-III transactions. A naïve approach is that for each time period, the sub-blockchain shares all newly added blocks with the main-blockchain, and the main-blockchain does the same to share new blocks with all sub-blockchains. This approach is equivalent to the case that each sub/main-blockchain keeps tracks the whole system, which makes the system not scalable and inefficient.

To overcome this challenge, an RSA accumulator [27] is used to compress the shared status of a sub/main-blockchain to reduce the cost. The scheme comprises 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 status when a new block is added. For the genesis block, it calculates
$$v_1 \leftarrow g^{\mathit{hash}(\mathit{blk}_1||1)} \mod N,$$
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
$$v_i \leftarrow v_{i-1}^{\mathit{hash}(\mathit{blk}_i||i)} \mod N.$$
- `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 $\rho_1 \leftarrow \mathit{hash}(\mathit{blk}'||i)$ and $\rho_2 \leftarrow g^{\prod_{k=1}^{n} \mathit{hash}(\mathit{blk}_k||k)/\rho_1} \mod N$.
- `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 $\rho'_1 \overset{?}{=} \mathit{hash}(\mathit{bkl}'_i||i)$ 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 as part of the new block. 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. Since the accumulator is calculated using all existing blocks, it reflects the current status of the blockchain.

B. Cross Blockchain Transaction 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 interact with a prover who has the complete blockchain to check whether a transaction is included using `ProofGen` and `ProofVerf`. The situation is more complex for the edge blockchain architecture as all components are inter-locked. Without loss of generality, we consider the scenario a transaction tx_Q initialized in a 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 [27] 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, there is a lack of such a trusted third party. We utilize a distributed RSA parameter generation scheme given in [28] to overcome 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 an agreement on P.
- For peer $i = 1, \ldots, k$, 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 $i = 1, \ldots, k$, it computes $p_{i,j} = f_i(j), q_{i,j} = g_i(j), h_{i,j} = h_i(j)$ for $j = 1, 2, \ldots, k$. Peer i privately sends $(p_{i,j}, q_{i,j}, h_{i,j})$ to peer j that $i \neq j$.
- For peer $i = 1, \ldots, k$, 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 peers 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 [28] 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 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.

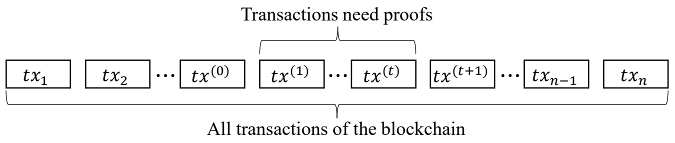

Fig. 3: Efficient batch proof generation. Transactions before and after the target set of transactions only need to be processed once to save computation cost.

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 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:

$$\begin{aligned}(g^{x_1x_2\ldots x_{i-1}x_{i+1}\ldots x_n})^{x_{i_1}} \quad \bmod N &\overset{?}{=} X_1 \Leftrightarrow \\ g_1^{x_{i_1}} \quad \bmod N &\overset{?}{=} X_1\end{aligned}$$

where $g^{x_1x_2\ldots x_{i-1}x_{i+1}\ldots x_n} = g_1$, 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:

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

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

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

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. Integration of Edge Blockchain with Distributed/Federated DNN Applications

Although there are different 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 edge blockchain with an IoT based DNN application to offer an end-to-end tracking, the system should satisfy three requirements:

- The data collection and training process are recorded as blockchain transactions, which are distributed to peers in the system.
- When the edge blockchain 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 the edge blockchain system.

A. Tracking of Data Collection

When edge computing is utilized to facilitate an IoT based DNN applications, IoT devices connect to different edge data centers based on their physical locations and submit collected data for local model construction, as depicted in Fig. 1.

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 $blkc^E$.
- Status 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.

Note that the originally collected data does not need to be stored on the corresponding sub-blockchain to improve the storage efficiency of the edge blockchain system.

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. Tracking of Model Training Process

Distributed/federated DNN training has received extensive attention [29], [30], [31], which fits the edge computing environment very well, i.e., IoT devices send data to connected edge data centers to build local models and the local models are merged in the cloud to obtain the final model.

Since the final model depends on multiple local models trained in edge data centers, we first describe tracking 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 local 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 local model generated in the previous round, and outputs a new intermediate local model. More formally, let $(d_1, d_2, \ldots)$ denote the sequence of data sets collected by local IoT devices for local model training, $\mathcal{M}_i$ denote the current local model generated by the training algorithm $\mathcal{T}$, and $\mathcal{M}_O$ denotes the aggregated model received from the cloud, we have

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

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.

Recording local model training process with the sub-blockchain. The local 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 protocol. $\mathcal{M}_i$ and d_{i+1} are local and their integrity is easy to verify using information stored in 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 so it is easy for the peer to obtain inputs and store outputs.
  - 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 of the local model to the 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 then applied to disseminate information of the transaction to the whole edge blockchain system. The final model training process in the cloud is tracked in the same way.

Verification of the model training process. The purpose of storing the 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 repeats 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 $\mathcal{M}_i$.

VI. Implementation and Experiments

In this section, we discuss the implementation of the proposed distributed/federated DNN application tracking mechanism and provide preliminary performance evaluation. We utilize the endorse-ordering based consensus to build the edge blockchain, where a transaction needs to be endorsed properly first and then added to the blockchain by ordering service.

A. Performance in the Normal Situation

We first evaluate the performance of using the edge blockchain to track the DNN construction in the normal situation where there is no attack. Transactions are proved by the system with a 2-of-any endorsement policy, and then forwarded to ordering service to be added to the sub/main-blockchain. Each transaction is used to track an intermediate training result or

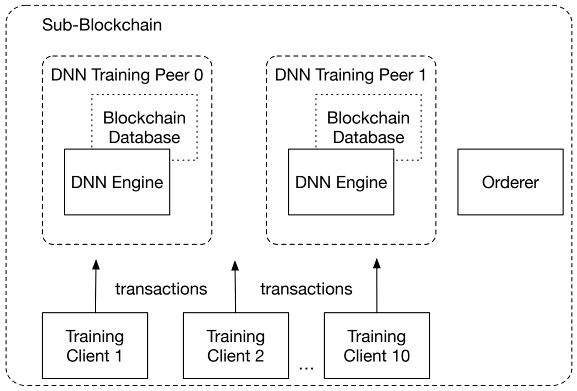

Fig. 4: Overview of the sub-blockchain prototype. During benchmarking, all transactions are driven by training clients, which are implemented using Hyperledger Fabric gateway [32]. A training client collects data from IoT devices and submits it to the training peers. Peers then train the local model with the data, convert the result to a hash tag, and store it in the blockchain. In this experiment, two training peers are involved in a sub-blockchain. The main-blockchain architecture is similar to the sub-blockchain.

a collected data set, and stored 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 done with an AWS EC2 t2.2xlarge instance, which has 8 vCPUs, 32 GB memory and 30 GB SSD vDisk. TABLE I shows the evaluation results.

TABLE I: Performance evaluation for transaction creation.

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

We assume that the transaction size is close to the hash size, which is generated from the intermediate training results/collected data. 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. The possible reason is that a larger transaction is more expensive for endorsement operation. 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 or trunked hash value.

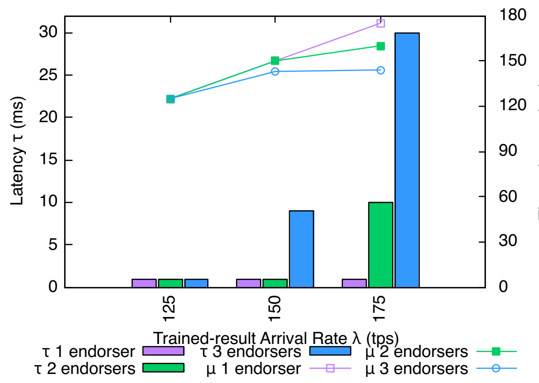

Fig. 5: Performance evaluation of edge blockchain-based DNN tracking using an *AND* or *OR* policy. Here λ is the number of trained results submitted to blockchain at a second, τ is the latency of the trained results being stored by the blockchain, and μ is the throughput of the trained results processing.

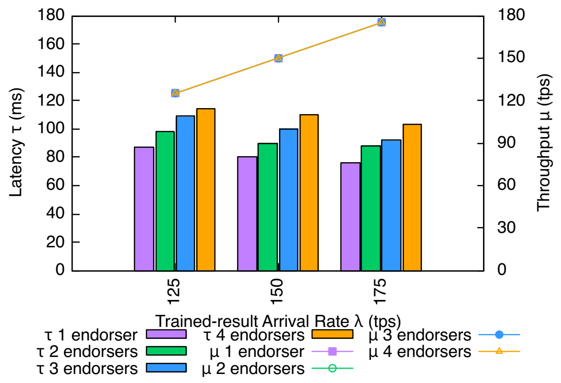

Fig. 6: Performance Evaluation of Blockchain-based DNN under *nOFANY* policy.

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, and vice versa. The performance of accumulator propagation is mainly affected by network delay. Because the size of the accumulator value is small, it only has a limited impact on the network delay.

B. Performance under Attack

A prototype edge blockchain using the endorsement-ordering consensus relies on endorsers, orderers, and CAs to provide secure tracking on the distributed platform. However, these computing nodes can be compromised by malicious users, and they may be utilized to attack the edge blockchain system. Generally, there are two ways to improve the security of the edge blockchain system: increasing the number of endorsers and optimizing the endorsement policy [33]. We consider the following basic endorsement strategies for the experiment: (i) *AND* policy. Require all endorsers in the list sign on the transaction. (ii) *OR* policy. Require at least one of endorsers in the list sign on the transaction. (iii) *nOFANY* policy. Require n of all the endorsers to endorse the transaction.

We conduct experiments and evaluate the performance under different configurations of endorsers and endorsement policy. To reduce the impact on running multiple endorsement peers in a single machine, this experiment is conducted on an AWS EC2 r6g.4xlarge instance, which has 16 vCPUs, 128 GB memory and 30 GB SSD vDisk.

From Fig. 5, we observe that when we have more endorsers with *AND* or *OR* policies, the latency increases and the throughput decreases. However, even with the results, we do not observe any significant impact on the performance, e.g. the latency can still maintain within 30 ms and the throughput only drops less than 25 tps approximately.

When the policy *nOFANY* is adopted (as depicted in Fig. 6), although the overall performance in terms of latency is not as good as a *AND* or *OR* policies, the throughput remains constant. In addition, it does not affect the performance significantly by adding more endorsers.

In summary, we can tweak some parameters of blockchain to balance the performance and the level of security offered by the edge blockchain to maintain the tracking information.

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 [34], [35], [36]. Biswas *et al.* proposed a blockchain scheme that is customized for IoT data protection [37], 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 [38]. This work aims at improving the performance of a single blockchain, which can be incorporated into the proposed edge blockchain 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 [39]. 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

Integration IoT and edge computing for distributed/federated DNN has many promising applications. There are mainly two types of attacks for such a system, data poisoning and model poisoning, which may cause serious consequences. Although there are existing works to detect and prevent such attacks, most of them are probabilistic and only work on the assumption that the input data does not change significantly. To mitigate the limitations, we propose to use the edge blockchain to offer an immutable and scalable tracking system for distributed/federated DNN applications, which provides an end-to-end integrity protection of the DNN application. The tracking information can also be used to identify and patch compromised components of the system. We also develop a prototype and conduct preliminary experiments to demonstrate the practicability of the proposed edge blockchain based distributed/federated DNN tracking mechanism.

References

[1] L. Holbrook and M. Alamaniotis, "A good defense is a strong dnn: Defending the iot with deep neural networks," in *Machine Learning Paradigms*. Springer, 2020, pp. 125–145.

[2] M. A. Al-Garadi, A. Mohamed, A. Al-Ali, X. Du, I. Ali, and M. Guizani, "A survey of machine and deep learning methods for internet of things (iot) security," *IEEE Communications Surveys & Tutorials*, 2020.

[3] A. Kanawaday and A. Sane, "Machine learning for predictive maintenance of industrial machines using iot sensor data," in *2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS)*. IEEE, 2017, pp. 87–90.

[4] R. Dhall and V. Solanki, "An iot based predictive connected car maintenance." *International Journal of Interactive Multimedia & Artificial Intelligence*, vol. 4, no. 3, 2017.

[5] H. H. Nguyen, F. Mirza, M. A. Naeem, and M. Nguyen, "A review on iot healthcare monitoring applications and a vision for transforming sensor data into real-time clinical feedback," in *2017 IEEE 21st International Conference on Computer Supported Cooperative Work in Design (CSCWD)*. IEEE, 2017, pp. 257–262.

[6] P. S. Pandey, "Machine learning and iot for prediction and detection of stress," in *2017 17th International Conference on Computational Science and Its Applications (ICCSA)*. IEEE, 2017, pp. 1–5.

[7] J. Tang, D. Sun, S. Liu, and J.-L. Gaudiot, "Enabling deep learning on iot devices," *Computer*, vol. 50, no. 10, pp. 92–96, 2017.

[8] D. Sahoo, Q. Pham, J. Lu, and S. C. Hoi, "Online deep learning: learning deep neural networks on the fly," in *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, 2018, pp. 2660–2666.

[9] 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.

[10] 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.

[11] Y. Chen, Y. Ning, and H. Rangwala, "Asynchronous online federated learning for edge devices," *arXiv preprint arXiv:1911.02134*, 2019.

[12] "Local model poisoning attacks to byzantine-robust federated learning," in *29th USENIX Security Symposium (USENIX Security 20)*. USENIX Association, Aug. 2020, pp. 1605–1622.

[13] R. Tomsett, K. Chan, and S. Chakraborty, "Model poisoning attacks against distributed machine learning systems," in *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, vol. 11006. International Society for Optics and Photonics, 2019, p. 110061D.

[14] G. R. Mode, P. Calyam, and K. A. Hoque, "False data injection attacks in internet of things and deep learning enabled predictive analytics," *arXiv preprint arXiv:1910.01716*, 2019.

[15] J. Zhang, J. Chen, D. Wu, B. Chen, and S. Yu, "Poisoning attack in federated learning using generative adversarial nets," in *2019 18th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/13th IEEE International Conference on Big Data Science and Engineering (TrustCom/BigDataSE)*. IEEE, 2019, pp. 374–380.

[16] R. Izmailov, S. Sugrim, R. Chadha, P. McDaniel, and A. Swami, "Enablers of adversarial attacks in machine learning," in *MILCOM 2018-2018 IEEE Military Communications Conference (MILCOM)*. IEEE, 2018, pp. 425–430.

[17] G. F. Cretu, A. Stavrou, M. E. Locasto, S. J. Stolfo, and A. D. Keromytis, "Casting out demons: Sanitizing training data for anomaly sensors," in *2008 IEEE Symposium on Security and Privacy (sp 2008)*. IEEE, 2008, pp. 81–95.

[18] O. Suciu, R. Marginean, Y. Kaya, H. Daume III, and T. Dumitras, "When does machine learning {FAIL}? generalized transferability for evasion and poisoning attacks," in *27th {USENIX} Security Symposium ({USENIX} Security 18)*, 2018, pp. 1299–1316.

[19] J. Feng, H. Xu, S. Mannor, and S. Yan, "Robust logistic regression and classification," in *Advances in neural information processing systems*, 2014, pp. 253–261.

[20] J. So, B. Guler, and A. S. Avestimehr, "Byzantine-resilient secure federated learning," *arXiv preprint arXiv:2007.11115*, 2020.

[21] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[22] 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.

[23] 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.

[24] F. J. Provost and D. N. Hennessy, "Scaling up: Distributed machine learning with cooperation," in *AAAI/IAAI, Vol. 1*. Citeseer, 1996, pp. 74–79.

[25] S. Savazzi, M. Nicoli, and V. Rampa, "Federated learning with cooperating devices: A consensus approach for massive iot networks," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4641–4654, 2020.

[26] 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.

[27] 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.

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

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

[30] 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.

[31] 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.

[32] 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.

[33] P. Thakkar, S. Nathan, and B. Vishwanathan, "Performance benchmarking and optimizing hyperledger fabric blockchain platform," *arXiv preprint arXiv:1805.11390*, 2018.

[34] 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.

[35] 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.

[36] 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.

[37] 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.

[38] 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.

[39] 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.