1-2020

DIoTA: Decentralized Ledger based Framework for Data Authenticity Protection in IoT Systems

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

Lin Chen
University of Houston

Zhimin Gao

Xinxin Fan

Taeweon Suh

See next page for additional authors

Recommended Citation

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," in IEEE Network, vol. 34, no. 1, pp. 38-46, January/February 2020, doi: 10.1109/MNET.001.1900136.

DIoTA: Decentralized Ledger based Framework for Data Authenticity Protection in IoT Systems

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

***Abstract*—It is predicted that more than 20 billion IoT devices will be deployed worldwide by 2020. These devices form the critical infrastructure to support a variety of important applications such as smart city, smart grid, and industrial internet. To guarantee these applications to work properly, it is imperative to authenticate these devices and data generated from them. Though digital signature can be applied for these purposes, the scale of the overall system and the limited computation capability of IoT devices pose two big challenges. In order to overcome these obstacles, we propose DIoTA, a novel decentralized ledger based authentication framework for IoT devices. DIoTA uses a two layer decentralized ledger architecture together with lightweight data authentication mechanism to facilitate IoT devices and data management. We also analyze the performance and security of DIoTA, and explicitly give the major parameters an administrator can choose to achieve the desirable balance between different metrics.**

I. Introduction

Internet of things (IoT) devices are fundamental building blocks in many applications of the fourth industrial revolution such as self-driving vehicles, advanced manufacturing systems, agriculture, and smart cities. The combination of AI and IoT (AIoT) plays a more critical role since it is able to provide improved human-machine interactions, enhanced data management and analytics, and more efficient management of IoT devices. The effectiveness of AIoT heavily depends on the data collected by IoT devices. Thus, the security should be essentially taken into consideration when building IoT systems. If an adversary can alter the input data to the AI module, he/she can influence the AI and lead to conclusions that may cause serious consequences. Therefore, it is critical for the system to guarantee the authenticity of the data collected by IoT devices. In other words, the end user should be able to verify that the data is generated by the expected IoT device and not compromised.

A modern IoT device is usually equipped with a public/private key pair and able to handle simple asymmetric cryptography operations such as the digital signature generation/verification, which can be used for data authenticity protection. Specifically, the IoT device can sign its generated data using its private key and send the data together with the signature to the data consumer (e.g., the model training component). Then, the consumer can verify the received data against the digital signature and corresponding public key. However, this approach is far away from satisfaction to be applied in IoT applications because of two reasons: (i) An IoT system usually involves a large number of devices, and it is challenging to manage all public key certificates in an efficient manner. (ii) An IoT device usually has limited computation capability and power supply. Therefore, it cannot afford frequent computation intensive asymmetric cryptography operations, especially when the devices relying on battery are deployed in the field. The situation becomes even more complex when IoT devices are owned and managed by multiple parties and the system relies on all the devices to operate. In this case, a signature based authenticity protection mechanism may not work as the owner can remove/alter the data generated by managed IoT devices without being detected by others.

The emerging decentralized ledger technology sheds light and provides a new way to protect the authenticity of the data collected by IoT devices in a collaborative environment. Decentralized ledger was first developed to build cryptocurrency schemes without relying on a trusted third party [1], and later it finds many other applications. In a nutshell, a decentralized ledger is a data structure maintained by multiple parties through a consensus protocol, and each party keeps its local copy of the ledger. It exhibits several enticing characteristics towards the data authenticity protection for IoT devices, such as immutability, high availability, and collaboration support with multiple parties.

Nevertheless, it is non-trivial to utilize the decentralized ledger for the authenticity protection due to the two reasons discussed earlier. In order to close the gap, we propose DIoTA, a decentralized ledger based framework for IoT data authenticity protection, which overcomes the challenges with two novel ideas: (i) *Layered decentralized ledger architecture.* To handle a large number of IoT devices and provide timely services, DIoTA adopts the divide-and-conquer strategy and proposes an edge-global architecture to organize multiple decentralized ledgers. In DIoTA, each edge ledger only serves a subset of IoT devices and the global ledger connects all edge ledgers to facilitate occasional cross-ledger data exchange. To reduce the cost and latency of cross-ledger information exchange/verification, DIoTA uses a novel cryptography accumulator based approach that allows one edge ledger to quickly verify whether a record on another edge ledger is valid or not. (ii) *Lightweight backward-forward secure data authenticity protection scheme using decentralized ledger.* To reduce the computation burden and save energy of IoT devices, DIoTA leverages the immutability feature of decentralized ledger to provide a lightweight backward-forward secure data authentication mechanism. Even if an adversary gets a temporary secret key for data authentication, he/she cannot tamper with previous or future data. The mechanism also minimizes the asymmetric cryptography operations on computation and power-constrained IoT devices.

We provide detailed performance and security analysis of DIoTA. The relationship among different parameters is also presented so that users can choose appropriate configurations for their unique requirements of IoT systems. To demonstrate the practicality, we also describe the implementation and utilize the experimental data to show its performance.

II. System Overview and Security Assumptions

Tis section presents an overview of the DIoTA framework and its design goals.

A. Decentralized Ledger in DIoTA

There are mainly two types of decentralized ledgers: public ledger and permissioned ledger. A public ledger allows anyone to participate in its maintenance. A user selects his/her own public/private key pair according to the system requirements (e.g., type and length of the key pair) by him/herself, and uses the private key to sign transactions and constructed blocks. Other users can verify a submitted transaction/block using the corresponding public key while they cannot learn the owner's information directly from the public key. A permissioned decentralized ledger also utilizes public/private key pairs to identify users in the system, but requires all public keys are recognizable, meaning that one can check whether a specific public key is part of the system and who is the owner of it. This is usually achieved through a public key infrastructure (PKI).

In DIoTA, we assume the ledgers are maintained by nodes which are controlled by entities that are involved in the system, such as owners of IoT devices which are responsible for data collection, data analytic service providers, and end users of the analytic results. Since these entities know each other, we consider permissioned decentralized ledger in DIoTA. In other words, nodes participating the decentralized ledger maintenance can recognize each other's messages by verifying digital signatures with public key certificates.

Although ledgers in DIoTA are maintained by authorized nodes, it does not mean that information managed by DIoTA is only accessible by these nodes. Depending on the use case, DIoTA may allow the public to read information stored in the decentralized ledgers while preventing them from updating them.

B. Overview of DIoTA

In DIoTA, there are three types of participants, and Fig. 1 shows an overview of the system.

IoT devices. Each IoT device is equipped with a unique public/private key pair and the public key is certified by a CA, and it uses this key pair to authenticate itself to other participants in the system. For the security guarantee, a variety of methods have been developed, from silicon level (e.g., anti-tamper technology) to software level (e.g., trusted execution environment) [2]. Since IoT devices are typically targeted for specific functionalities, it is relatively uncomplicated to apply these technologies than devices with complexities. DIoTA is orthogonal to these methods as it aims at protecting the authenticity of the generated data from IoT devices. Therefore, we assume the IoT device is trusted in this work. That is, it follows pre-defined protocols and is able to safely store and use the secret information such as the private key.

Decentralized ledger nodes. These nodes work together to maintain the ledgers used in DIoTA, which provide IoT data authenticity protection and other related services. We do not fully trust a single node as it can be compromised by an adversary. However, it is very unlikely that an adversary can take over the majority of these nodes at the same time, and the underlying consensus mechanism guarantees that the decentralized ledger as a whole is trusted and preserves all desirable features including immutability and correctness of smart contract execution.

Cloud and edge servers. Cloud and edge servers are powerful devices in terms of computation capability and storage capacity. They cooperate to store the data generated by connected IoT devices and help the devices to communicate with each other. Cloud and edge servers are not fully trusted. An adversary may be able to alter IoT devices generated data. When such an event occurs, the DIoTA helps the end users detect the modification.

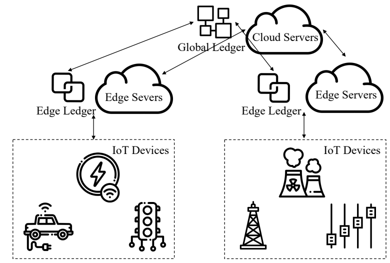

Fig. 1: Overview of DIoTA and its relationship with edge-cloud infrastructure. DIoTA is composed of two types of decentralized ledgers (edge ledger and global leger), which work together to provide IoT data authenticity protection service. Edge servers and cloud servers collaborate to provide storage and communication services to IoT device.

Integrating cloud and edge computing technologies to serve IoT systems has been studied extensively [3], [4]. This paper focuses on the design of the decentralized ledger structure and the interaction protocol between IoT devices and the ledger. DIoTA can be easily integrated with existing IoT-edge-cloud design. Notations used in the description of DIoTA are summarized as follows: (i) $\mathcal{D}$ is the set of all IoT devices. (ii) $\mathcal{D}_i$ is the ith IoT devices subset and $\mathcal{D}_i \subset \mathcal{D}$. d_{ij} is the jth IoT device in $\mathcal{D}_i$. We also assume $\cup_{i=1} D_i = \mathcal{D}$ and $D_i \cap D_j = \emptyset, i \neq j$. (iii) EL_i represents both the ith edge ledger and the set of nodes that maintain EL_i. $el_{ij} \in EL_i$ is a node of the edge ledger. (iv) GL represents both the global ledger and the set of nodes maintain GL, $gl_i \in GL$ is a node of the global ledger.

C. Design Goal and Objectives of DIoTA

The goal of DIoTA is to protect the authenticity of data collected by IoT devices. Considering the characteristics of IoT system, the design goal is further divided into following objectives:

- *Performance*. DIoTA should be able to support a large number of IoT devices and reduce the computation burden off from those devices.
- *Security*. When the underlying decentralized ledger is secure (i.e., the decentralized ledger as a whole behaves honestly and follows the pre-defined protocols), DIoTA should be able to detect breakages when adversaries manipulate the data from IoT devices and guarantee the data authenticity.
- *Management*. DIoTA should allow IoT devices to join and/or leave the system. It also should allow them to move from one place to another without the protection disruption.

III. Detailed Design of the DIoTA Framework

This section details the DIoTA framework for the IoT data authenticity protection.

A. Connection of Different Components

The set of IoT devices $\mathcal{D}$ are divided into groups D_i, and each group D_i is served by an edge ledger EL_i, which is maintained by multiple nodes e_{ij} (edge servers). These edge ledger nodes can be deployed on the edge and they are usually close to the IoT devices they serve. There is also a global ledger GL, which is maintained by a group of nodes that usually sit in the cloud. Each edge ledger EL_i is connected to the global ledger *GL*, but different edge ledgers are not directly connected with other in order to reduce the system complexity.

Note that an edge ledger is a decentralized system and the IoT device d_{ij} can connect to an arbitrary node e_{ij} of the corresponding edge ledger EL_i. To tolerate the failure of edge ledger nodes, the IoT device d_{ij} locally keeps a list of nodes of EL_i. When d_{ij} tries to connect to EL_i, it iterates through the list of nodes until succeeded. If one or more nodes in the current list are not accessible, d_{ij} updates its local list by replacing them with new nodes with the help of the edge ledger node that it has been successfully connected to.

An edge ledger and the global ledger can share some nodes for connection (i.e., $GL \cap EL_i \neq \emptyset$). The implementation detail is presented in Section V.

B. Certificate Management

Each IoT device $d_{ij} \in D_i$ has a certificate $cert_{ij}$, which is generated by a CA and registered with EL_i. EL_i embeds $cert_{ij}$ in a transaction, and then includes it to a block of the ledger maintained by nodes of EL_i. Therefore, when the device d_{ij} signs a message using its private key, nodes in EL_i can verify its validity with the corresponding public key certificate. Considering the large number of IoT devices, EL_i does not actively share its managed certificates with other edge ledgers and the global ledger.

Each edge/global ledger node also has its own certificate to support the decentralized ledger operations. Unlike IoT devices' certificates, all ledger nodes' certificates are stored in each edge ledger and the global ledger, and all ledger nodes can authenticate each other using digital signatures and stored certificates. Compared with the number of IoT devices, the total number of edge/global nodes is limited. Thus, keeping a copy of their certificates in each ledger is not a big burden.

A certificate revocation is performed by generating a revocation transaction on the corresponding ledger(s). The revocation of an IoT device's certificate is managed by the edge, whereas the revocation of a ledger node's certificate is broadcasted to all the ledgers in the system. When the cross-edge-ledger verification is required, the protocol described in Section III-F is utilized to enable efficient information verification between different edge ledgers, which only requires exchanging of a constant amount of data.

C. IoT Device Authentication

The device authentication is the prerequisite for the data authenticity protection. An IoT device is authenticated using digital signatures generated with its private key, and one can verify it with its certificate. DIoTA provides several services to facilitate the device authentication:

- *Signature verification.* An IoT device d_{ij} submits its digital signature to the connected edge ledger EL_i, and nodes in EL_i can verify the signature with the certificate stored on the edge ledger.
- *Certificate checking.* In case the IoT device d_{ij} just moves to a new edge ledger $EL_{i'}$, $EL_{i'}$ obtains its certificate from the original edge ledger EL_i with the help of the global ledger using the efficient information exchange protocol given in Section III-F.

D. IoT Data Authentication

There are two major interaction models in IoT communication protocols: the request-reply model and publish-subscribe model. For both cases, SSL can be utilized to protect the data authenticity in transmission. However, the energy required to maintain the SSL connection is not negligible for IoT devices. Furthermore, SSL is not able to protect the authenticity of data at the rest. Therefore, a customized IoT data authenticity protection should be provided under the DIoTA framework.

Data authenticity protection schema. DIoTA uses data authenticity protection schemas to manage information needed for data authenticity protection and these schemas are maintained by edge ledgers other than the IoT devices. Therefore, an IoT device does not need to manage sessions by itself. A schema consists of 6 fields: (i) *Sender.* Certificate of the IoT device sending data. (ii) *Data unit.* The unit of data transmission, which can be 1 KB, 1 MB, or other sizes based on the preference of the IoT device. (iii) *Data authentication mechanism.* The IoT device can select different ways to authenticate generated data, such as HMAC and CMAC. (iv) *Key information.* The key used in the selected data authentication mechanism. (v) *Key updating frequency.* After the authentication key is used for a certain number of data units, the IoT device sending data will update the key and everyone should stop using the old key for new data units' authenticity. (vi) *Lifespan of the schema.* This field is an integer that determines how many times the authentication key can be updated. Each IoT device can select its own parameters for the schema according to its special requirements.

The *key information* field is hidden from the public when created, to prevent an adversary from generating valid MACs for fake data, but the key needs to be disclosed later to allow others to use it to verify the sender generated MACs.

As information stored in this field is intended to be disclosed later, DIoTA uses a cryptographic commitment scheme instead of encryption to temporarily hide the key. A cryptographic commitment scheme has two basic features, hiding and binding. The hiding feature prevents an adversary from learning the committed value by observing the commitment, and the binding feature guarantees that no one can modify the committed value after the commitment is generated and published. In order to reduce the computation cost of IoT devices, DIoTA uses the HMAC based commitment scheme. To commit a value *val*, the IoT device randomly selects a secret key *hkey*, and computes the commitment as $ci \leftarrow \texttt{HMAC}(val, hkey)$. `HMAC` is built based on a cryptographic hash function, and we use `SHA256` in this work and set the length of *hkey* as 256 bits. The *key updating frequency* field determines how long an authentication key can be used. It also affects the waiting time for one to verify the authenticity of recently uploaded data, as it only obtains the key after the key is updated to a new version.

The device d_{ij} submits its data authentication schema together with a digital signature to the edge ledger EL_i for verification and storage. The end user of the data can obtain the schema information from EL_i. If the end user is not connected to EL_i directly, it can ask its connected edge ledger to obtain the schema information using the efficient information exchange protocol in Section III-F.

Data authentication key updating and verification. To reduce the computation cost for IoT devices, DIoTA minimizes the use of expensive public key cryptography operations. In a nutshell, DIoTA allows an IoT device to generate a single digital signature for a data authentication schema which can be used for a relatively long time and large amount of data by updating the *key information* field multiple times. After a key updating, the previous key is released to others for the data authenticity verification, and the IoT device uses the new key

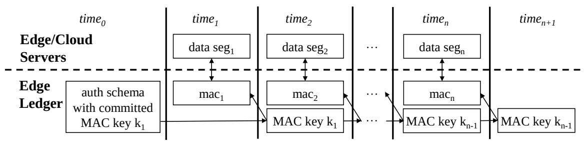

Fig. 2: The process for an IoT device to send data is divided into discrete steps, and the IoT device waits for one step to be finalized before starting the next one. The device submits the authentication schema before sending real data, and discloses the secret MAC key used for previous data segment together with the new data segment. The open of the first MAC key k_1 requires the open operation of the commitment scheme. In this work, the IoT device reveals k_1 together with the randomly selected HMAC key to open the commitment of k_1.

to protect next data segments. To implement such a scheme, the key updating algorithm needs to satisfy two requirements:

- Backward-forward security. Given all existing keys, an adversary should not be able to learn future keys that have not been disclosed by the IoT device and he/she cannot tamper historical data protected by existing keys.
- Public verifiability. Given all existing authentication keys and a new key, everyone should be able to verify whether the new key is generated by the legitimate IoT device.

DIoTA uses reversed hash list for the key update to meet these two requirements. When building the data authenticity protection schema, the IoT device determines the *lifespan* field, which is a positive integer n that specifies how many times it can update the data authentication key with this schema, and selects a random MAC key k_n, e.g., a 256 bits string. The device then computes:

$$k_{n-1} \leftarrow \mathtt{H}(k_n||n-1), \cdots, k_1 \leftarrow \mathtt{H}(k_2||1), \qquad (1)$$

where $\mathtt{H}(\cdot)$ is a cryptographic hash function with an adequate output size, e.g., `SHA256`. The IoT device uses k_1 as the first authentication key and includes the commitment of k_1 in the data authentication schema. The device then updates the MAC key from k_{t-1} to k_t, $1 < t \leq n$ when necessary.

Data transmission. After the data authentication schema is accepted by an edge ledger, the IoT device can start to send data. The device uses two channels for the transmission: one for actual data and one for authenticity protection information, which is depicted in Fig. 2. Specifically, the actual data is sent to connected edge servers and stored together with cloud servers, while corresponding meta data including authenticity protection information is sent only to connected edge ledger. The actual data are divided into segments and the IoT device generates a MAC for each segment using a different key. When the $(t+1)^{\text{th}}$ segment is sent to edge/cloud servers, its MAC is also sent to the edge ledger and the sender discloses the MAC key of the segment t, which can be utilized to verify the corresponding t^{th}data segment. The last MAC key k_n is selected by the sender device, and other MAC keys are derived from k_n using the key updating method described above.

Data verification and storage. The data from IoT devices are collected and stored by edge/cloud servers and DIoTA is responsible for managing and verifying authenticity information. When the data segment seg_t is sent and stored by edge/cloud servers, the corresponding edge ledger cannot verify its authenticity immediately. Instead, the edge ledger waits for the corresponding authenticity protection key k_t, which is provided by the IoT device in time period $t+1$. Then, each edge ledger node conducts the following checks with k_t:

- $mac_t \stackrel{?}{=} \mathtt{MAC}(seg_t, k_t), t \geq 1$, and
- $\mathtt{H}(k_t||t-1) \stackrel{?}{=} k_{t-1}, t > 1$.

Here k_{t-1} is the previous data authentication key and has been stored in the edge ledger. Edge ledger nodes then run the consensus protocol to determine whether to include k_t to the ledger. The data consumer (e.g., an AI model training module) obtains authenticity information from DIoTA and verifies whether the data is compromised or not.

E. Key Information Query

DIoTA maintains a large number of keys to facilitate IoT data authenticity protection. Each key is embedded in a transaction, and transactions are organized as blocks linked together. During the operation of DIoTA, a node needs to query the ledger to obtain corresponding keys for data authenticity validation and IoT device authentication. Since querying a key does not require modification of the ledger, these operations will not trigger the consensus mechanism which is usually more expensive. Furthermore, though all transactions/blocks are logically organized in a linear chain structure, DIoTA can utilize a database system to store them. The order information can be maintained as part of the database schema and queries can be efficiently handled.

F. Efficient Information Exchange between Edge Ledgers using the Global Ledger

Edge ledgers are not connected to each other directly, and they do not keep copies of other ledgers. Therefore, it is not trivial for an edge ledger to verify whether a block belongs to another edge ledger is valid. DIoTA provides a mechanism to exchange information efficiently and reliably in case the data user connected to one edge ledger needs to use data collected by IoT devices connected to another edge ledger. Without loss of generality, we assume EL_1 needs to get a transaction *tx* from EL_2 through the global ledger *GL* and verify its validity. The efficient information exchange protocol relies on a cryptographic accumulator and works as follows:

- EL_2 processes and accepts a new transaction *tx*:
  - EL_2 initializes the accumulator value *acc* when the edge ledger is set up.
  - *tx* is embedded into a block *blk*, and EL_2 updates the accumulator to acc';
  - All $el_{2j} \in EL_2$ run the consensus protocol to include both the new block *blk* and updated accumulator acc'.
- EL_2 updates its status to the global ledger *GL*:
  - To reduce the burden of *GL*, EL_2 only updates its status to *GL* after a certain number of new blocks are created, and only sends the accumulator value to *GL*;
  - *GL* checks whether EL_2 has achieved consensus on the updated accumulator acc', and includes the pair (EL_2, acc') into a new block if it passes the checking.
- EL_1 checks the validity of *tx*
  - EL_1 obtains the current accumulator value of EL_2 from the global ledger *GL*;
  - EL_1 requests a proof from EL_2 on the block *blk* that contains the transaction *tx*;
  - EL_2 responds to EL_1 with a proof that *blk* is included in edge ledger EL_2;
  - EL_1 verifies the proof and then utilizes the information stored in *tx*.

This protocol is used for multiple purposes in DIoTA, including retrieval of certificate information of an IoT device served by a different edge ledger, checking the authentication schema and MAC keys stored in another edge ledger.

IV. Analysis of DIoTA

In this section, we analyze the performance and security of DIoTA.

A. Performance Analysis of DIoTA

DIoTA uses data authenticity protection schemas to protect the authenticity of data collected by IoT devices. For an IoT device, it only uses a schema to protect a certain amount of data and then generates a new schema. Therefore, we analyze the operation complexity for each schema.

Cost of IoT devices. The IoT device itself needs to store a small amount of data such as its private key, cryptographic operation parameters, and list of edge ledger nodes. Thus, the storage is not demanding in IoT devices. We focus on the computation and communication cost of the IoT device in DIoTA.

- Preparation of data authentication schema. The IoT device needs to authenticate the schema to the edge ledger by generating a digital signature. The computation cost is one signature generation, and the communication cost is for transmitting one digital signature plus the schema itself.
- MAC keys generation. To build a schema, the IoT device pre-computes the reversed hash list, and the number of hash computation is determined by the *lifespan* field. Assuming *lifespan* is set to num_{key}, in order to submit the i^{th} data segment, the IoT device computes $num_{key} - i$ hashes on the originally MAC key if it does not store any intermediate results. Therefore, when the IoT device does not cache any previous computation results, the computation complexity is $\mathcal{O}(num_{key}^2)$ hash calculations, and the communication cost is for transmitting num_{key} hash values.
- MACs generation. The IoT device is responsible for generating MACs of collected data using MAC keys. The computation complexity of this operation is $\mathcal{O}(sz_{unt} \times num_{freq} \times num_{key})$, which is proportional to the amount of data that is protected by the given schema. Since only the computed MACs need to be sent out, the communication cost is for transmitting num_{key} MACs.

Cost of the edge ledger nodes. The complexity of operation of a decentralized ledger is usually measured by the number of transactions it has to process, and we use this metric to evaluate the performance of edge ledgers and the global ledger. An edge ledger mainly processes and stores three types of transactions: certificates of IoT devices, data authentication schemas, and MACs of IoT generated data. The edge ledger only creates certificates related transactions when a new IoT device is connected to the edge ledger or an existing certificate of an IoT device is revoked. The frequency of these two operations is usually very low and most transactions are related to the data authentication, i.e., data authentication schemas and MACs created by connected IoT devices. The number of such transactions is determined by following parameters: (i) num_d, the number of IoT devices connected to the edge ledger;

(ii) num_{key}, the number of keys that an authentication schema can support, which is the *lifespan of the schema* field. The number of transactions an edge ledger processes for a single data authenticity protection schema is about $num_d \times num_{key}$.

Cost of the global ledger nodes. The major function of the global ledger is to monitor and track the updates of each edge ledger to facilitate the information exchange between different edge ledgers and prevent edge ledgers from modifying historical data. Without loss of generality, we assume that each edge ledger is connected to the same number of IoT devices with the same data authenticity protection schema configuration and each device generates data at the same rate. The number of transactions the global ledger should handle is then determined by following parameters: (i) num_ℓ, the number of edge ledgers connected to the global ledger; (ii) num_{aggre}, the aggregation factor (a positive integer) that controls how often updates occur in the global ledger. The number of transactions the global ledger handles for a single data authenticity protection schema is $num_\ell \times num_d \times num_{key}/num_{aggre}$.

Latency of data authentication. Besides the latency caused by the decentralized ledger operations, the *key updating frequency* (denoted as num_{freq}) also affects the waiting time to verify the authenticity. This is because the sender releases the MAC key of a data segment after the whole segment is received by the edge ledger. A larger num_{freq} means that a single data authentication key is used to protect more data segments, while at the same time it also increases the latency as the sender needs more time to transmit the data.

Note that the block size also affects the performance of DIoTA but we ignore it in the analysis since it is a general parameter for all decentralized ledger based systems. With a large block size, one can usually expect higher throughput while it may also increase the latency.

B. Security Analysis of DIoTA

Under the framework of DIoTA, data generated by IoT devices are stored in edge/cloud servers, and DIoTA focuses on the protection of data authenticity.

Confidentiality of authenticity protection key. When an adversary learns an unused authenticity protection key, he/she can generate a valid MAC on arbitrary data. We focus on the confidentiality of the first authenticity protection key and the safety of other derived keys is guaranteed by the forward security feature. Before the first key is disclosed to the public, it is stored in the edge ledger as a commitment. As the IoT device keeps the commitment key as a secret, the hiding feature of the commitment scheme guarantees the confidentiality of the first authenticity protection key.

Forward security. Forward security requires that when a key in a key sequence is disclosed, the adversary cannot guess these keys that have not been disclosed. This feature is guaranteed by the one-wayness of the cryptographic hash function. Without loss of generality, we consider the case where $k_1, k_2, \ldots, k_t, t < num_{key}$ from an IoT device have been disclosed to the public. If there is an adversary $\mathcal{A}$ that can efficiently discover a data authentication key $k_{t'}, t < t' < num_{key}$, then $\mathcal{A}$ can efficiently reverse a function in the form of $\mathtt{H}'() = \mathtt{H}(\mathtt{H}(\cdots\mathtt{H}()))$, where $\mathtt{H}()$ is a secure cryptographic hash function that takes input with the same size of output. When we use $\mathtt{SHA256}$ as $\mathtt{H}()$ and the input size of $\mathtt{H}()$ is the same the output size, the composition of $\mathtt{H}()$ is equivalent to increasing the number of hash iterations, which does not affect the preimage resistance feature. Therefore, $\mathcal{A}$ cannot learn an unused authentication key as long as the underlying cryptographic hash function is preimage resistant and the original input size is equivalent to the output size of the hash function.

Backward security. Backward security requires that an adversary knowing all existing MAC keys cannot compromise historical data. This feature is guaranteed by the immutability feature of the decentralized ledger. Since all MAC keys and MAC values are finalized in blocks, the adversary cannot tamper the historical data unless he/she can take over the ledgers.

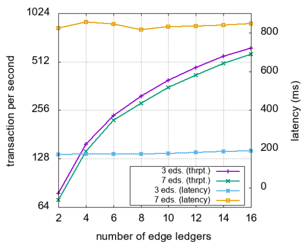

Fig. 3: Performance of transaction submission.

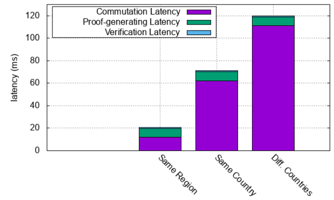

Fig. 4: Latency of query with nodes in different locations.

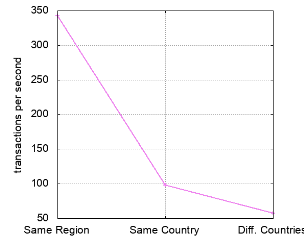

Fig. 5: Throughput of query with nodes in different locations.

V. Implementation and Experimental Data

This section describes the implementation of DIoTA and discusses the experimental results. The implementation of DIoTA consists of two parts, the IoT device part and the edge/global ledger part.

A. Device Side Implementation

Device side implementation is straightforward, we focus on three operations: generating digital signature on data authenticity protection schema, calculating hash function for initial key commitment and key updating, and computing CMAC [5] of generated data. Note that we do not use HMAC for data authentication as hash function is used to derive the sequence of data authentication keys, and using the same hash function for both key generation and data authentication may cause security problems. Some experimental results using ARM Cortex-M processor are summarized as follows [6]: (i) With curve secp256k1, it takes about 486 ms to generate a signature. (ii) To compute SHA-256 of 1024 bytes data, it takes 0.6 ms, which equals evaluation of 16 SHA-256 functions, and it costs about 0.0375 ms to evaluate a single SHA-256 function. (iii) To compute AES-CMAC of 1024 bytes of data with a 256 bits key, it takes 0.9 ms. Compared with digital signature operations, the hash and AES-CMAC-256 computation cost is almost negligible.

B. Edge/Global Ledger Implementation

We use Hyperledger Fabric [7] as the foundation to build edge ledgers and the global ledger. There are three types of nodes in Hyperledger Fabric that work together to process a transaction: peers, endorsers, and orderers. The life cycle of a transaction is as follows: (i) A peer initializes a transaction tx and sends it to all endorsers in the ledger. tx can be a data
from another edge ledger, and a response to a request. (ii) An endorser who receives tx checks it and signs the transaction using his/her private key if tx is valid. (iii) A peer collects all endorsements from endorsers and put them together with tx, which are then submitted to orderers. (iv) After an orderer receives tx with endorsements, it checks whether it satisfies the pre-defined endorsement policy (e.g., all listed endorsers need to sign), and then tries to pack it to a block and appends to the ledger. All orderers run the pre-defined consensus protocol to determine how a transaction should be added. The current implementation uses Kafka [8] as the consensus protocol. Each peer node keeps a copy of the ledger.

In DIoTA, edge ledgers and the global ledger are implemented as different *channels*, i.e., each edge ledger is a channel and the global ledger is also a channel. Channel is a concept of Hyperledger Fabric, which is a private subnet maintained by two or more specific network members for the purpose of conducting private and confidential transactions. Based on the original design of Hyperledger Fabric, different channels cannot talk with each other. This prevents an edge ledger from interacting with the global ledger. To overcome this challenge, we enroll one or more peers of an edge ledger to the global ledger. Each of these nodes keeps two ledgers at the same time. In order to support information exchange between an edge leger and the global ledger, we deploy a daemon on the node that can access both ledgers. From the storage perspective, each ledger is just a local database, and the daemon can conduct arbitrary operations on these ledgers. To reduce the complexity, the daemon only queries the two ledgers, and this does not affect the operation of the original Hyperledger Fabric system. We also implement the efficient information exchange protocol using RSA accumulator [9] with a modular size of 2048 bits and embed it in the cross-channel query daemon.

We conduct experiments on the developed prototype and focus on the following metrics:

- The overall throughput/latency of transaction processing. These metrics reflect how many transactions DIoTA can process at the same time and how fast it can process a transaction. The transactions include submission of the new authenticity protection schemas, key updating operations, and MACs uploading. Fig. 3 summarize the experimental results of throughput and latency respectively. We observe that the number of endorsers does not affect the performance significantly.
- The latency/throughput of cross-channel queries processing. These two metrics reflect the efficiency of cross-channel query operations when the data user is served by an edge ledger that is different from the edge ledger that serves the IoT devices collecting the data. Fig. 5 and Fig. 4 demonstrates the performance in different scenarios. The experimental data is collected based on a minimum cross-channel query requirement which only involves two edge ledgers and the global ledger. The latency of such operations is dominated by communication and the physical locations of the nodes also affect the performance significantly.

Note that we do not adapt any optimization to the Hyperledger Fabric system itself in our prototype, which is orthogonal to this work and has been extensively studied [10]. By using techniques such as in-memory database and reducing message size for endorsing/ordering, the performance of DIoTA can be further improved.

VI. Related Works

In this section, we briefly review related works on using decentralized ledger for IoT data protection and compare DIoTA with them.

Li et.al. proposed an efficient digital signature scheme which allows an IoT device to sign once for data generated in a

TABLE I: Comparisons of IoT data authenticity schemes.

	scalability	lightweight computation on IoT side	authenticity of data at rest	collaborative environment support
[11]	na	✓	✗	✗
[12]	✗	✓	✓	✓
[13]	✗	✗	✓	✓
[14]	✗	✗	✓	✓
DIoTA	✓	✓	✓	✓

given epoch [11], the goal of which is similar to the data authenticity protection schema of DIoTA. But this work does not consider integration with decentralized ledger, and does not enjoy all the desirable features. Shafagh et.al. designed a scheme that leverages blockchain to provide an auditable IoT storage service, which also supports protection of data authenticity [12]. This work treats blockchain as a black box and does not consider the scalability issue to support a large number of IoT devices. Liu et.al. developed an IoT data authenticity protection scheme that stores encrypted hash values of generated data in the decentralized ledger [13]. Though it achieves a similar goal, it does not consider the scalability of the underlying decentralized ledger neither, and the computation/power limitations of IoT devices are ignored. Machado et.al. adopted a three layer structure to manage IoT generated data [14], which is similar to our work. But unlike DIoTA that uses multiple edge ledgers to serve a large number of devices, their approach includes three relatively independent decentralized ledgers running at different layers, which does not help on scalability.

TABLE I summarizes the comparison between DIoTA and other IoT data authenticity protection mechanisms, especially those utilizing decentralized ledger technology.

VII. Conclusion

IoT has become an important information collection infrastructure for a variety of applications. As the collected information is used to support decision making and operations, it is critical to protect the data authenticity. Although digital signature can be used for device and data authentication, it does not fit the IoT scenario well because of the large number of devices in the system and limited device computation/power capacity. In order to fill the gap, DIoTA integrates a novel data authenticity protection mechanism with a layered decentralized ledger structure to achieve system scalability and reduction of computation burden on IoT devices at the same time. We implement a prototype of DIoTA using the Hyperledger Fabric codebase and also evaluate its performance.

Acknowledgement

This work was partially supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No.2019-0-00533, Research on CPU vulnerability detection and validation).

Lei Xu. Lei Xu received his Ph.D. degrees in 2011 from Chinese Academy of Sciences, China. He is an assistant professor at the Department of Computer Science, University of Texas Rio Grande Valley. His research interests include applied cryptography, cloud/mobile security, and decentralized systems.

Lin Chen. Lin Chen received his Ph.D. from Zhejiang University in 2013. He is an assistant professor at Department of Computer Science, Texas Tech University. His research interests include algorithms and complexity, distributed systems and cybersecurity.

Zhimin Gao. Zhimin Gao received his Ph.D. of Computer Science from University of Houston. He is currently an assistant professor at Auburn University at Montgomery. His research interests include blockchain, cloud computing, cybersecurity, IoT, and 5G.

Xinxin Fan. Xinxin Fan received the Ph.D. degree in electrical and computer engineering from the University of Waterloo, Canada, in 2010. He is the Head of Cryptography at IoTeX. His research interests range from fast and secure software and hardware implementations of cryptographic algorithms to the design and analysis of security protocols for communication networks, embedded systems, cloud computing and blockchain. He is a member of IEEE.

Taeweon Suh. Taeweon Suh received his B.S. degree from Korea University in 1993, M.S. degree from Seoul National University in 1995, Korea, and Ph.D. degree from the Georgia Institute of Technology in 2006, USA. He is a Professor in the Department of Computer Science and Engineering, Korea University. His research interests include hardware security, AI accelerators, and reconfigurable and embedded systems. Taeweon Suh is the corresponding author.

Weidong Shi. Weidong Shi received his Ph.D. of Computer Science from Georgia Institute of Technology where he did research in computer architecture and computer systems. Currently, he is employed as an associate professor by University of Houston.

References

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.
[2] ARM, "Iot security handbook."
[3] B. Du, R. Huang, Z. Xie, J. Ma, and W. Lv, "Kid model-driven things-edge-cloud computing paradigm for traffic data as a service," *IEEE Network*, vol. 32, no. 1, pp. 34–41, 2018.
[4] W. Zhao, J. Liu, H. Guo, and T. Hara, "Etc-iot: Edge-node-assisted transmitting for the cloud-centric internet of things," *IEEE Network*, vol. 32, no. 3, pp. 101–107, 2018.
[5] J. Song, R. Poovendran, J. Lee, and T. Iwata, "The aes-cmac algorithm," Tech. Rep., 2006.
[6] H. Tschofenig and M. Pegourie-Gonnard, "Performance of state-of-the-art cryptography on arm-based microprocessors," in *NIST Lightweight Cryptography Workshop*, vol. 2015, 2015.
[7] 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*. ACM, 2018, p. 30.
[8] J. Kreps, N. Narkhede, J. Rao *et al.*, "Kafka: A distributed messaging system for log processing," in *Proceedings of the NetDB*, 2011, pp. 1–7.
[9] 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.
[10] C. Gorenflo, S. Lee, L. Golab, and S. Keshav, "Fastfabric: Scaling hyperledger fabric to 20,000 transactions per second," *arXiv preprint arXiv:1901.00910*, 2019.
[11] X. Li, H. Wang, Y. Yu, and C. Qian, "An iot data communication framework for authenticity and integrity," in *2017 IEEE/ACM Second International Conference on Internet-of-Things Design and Implementation (IoTDI)*. IEEE, 2017, pp. 159–170.
[12] H. Shafagh, L. Burkhalter, A. Hithnawi, and S. Duquennoy, "Towards blockchain-based auditable storage and sharing of iot data," in *Proceedings of the 2017 on Cloud Computing Security Workshop*. ACM, 2017, pp. 45–50.
[13] B. Liu, X. L. Yu, S. Chen, X. Xu, and L. Zhu, "Blockchain based data integrity service framework for iot data," in *2017 IEEE International Conference on Web Services (ICWS)*. IEEE, 2017, pp. 468–475.
[14] C. Machado and A. A. M. Fröhlich, "Iot data integrity verification for cyber-physical systems using blockchain," in *2018 IEEE 21st International Symposium on Real-Time Distributed Computing (ISORC)*. IEEE, 2018, pp. 83–90.