2017 IEEE 24th International Conference on Web Services

Blockchain based Data Integrity Service Framework for IoT data

Bin Liu*[‡][§], Xiao Liang Yu[§], Shiping Chen[‡][§], Xiwei Xu[§] and Liming Zhu[‡][§]
[‡]*School of Computer Science and Engineering, University of New South Wales, Australia*
[§]*CSIRO Data61, Australia*
**Email: bin.liu@data61.csiro.au*

***Abstract*—It is a challenge to ensure data integrity for cloud-based Internet of Things (IoT) applications because of the inherently dynamic nature of IoT data. The available frameworks of data integrity verification with public auditability cannot avoid the Third Party Auditors (TPAs). However, in a dynamic environment, such as the IoT, the reliability of the TPA-based frameworks is far from being satisfactory. In this paper, we propose a blockchain-based framework for Data Integrity Service. Under such framework, a more reliable data integrity verification can be provided for both the Data Owners and the Data Consumers, without relying on any Third Party Auditor (TPA). In this paper, the relevant protocols and a subsequent prototype system, which is implemented to evaluate the feasibility of our proposals, are presented. The performance evaluation of the implemented prototype system is conducted, and the test results are discussed. The work lays a foundation for our future work on dynamic data integrity verification in a fully decentralized environment.**

***Keywords*-Blockchain; IoT; Data Integrity Verification**

I. Introduction

We have entered Big data age, businesses and individuals are collecting data at unprecedented scale, especially after IoT gained popularity [15]. Big data analysis drives almost every aspect of modern society [14]. There is a high demand for securely processing big data with low cost and high efficiency [9]. Various cloud computing services emerged in recently years and quickly became popular for big data processing, especially those big data generated by numerous IoT devices [10]. Cloud Storage Service (CSS) is among the most critical ones of them. Most of the Cloud Service Providers (CSPs) offer flexible approaches for Data Owners (DOs) to manage their data sets stored remotely in the cloud. Therefore, there have been plenty of IoT applications choose to store and process data on the cloud. Cloud Storage Service (CSS), however, is not secure by nature. In terms of security, there has been increasing demand for managing cloud storage service at equivalent security levels to enterprise storage systems. Cloud users typically have no control over the cloud storage servers being used, which means there are risks of Data Confidentiality, Data Integrity, and Data Availability. In our research, Data confidentiality means the data is only exposed to authorized parties. Data integrity is the maintenance of, and the assurance of the accuracy and consistency of, data over its entire life-cycle. Data availability concerns whether data can be accessed by authorized users anytime and anywhere. The cloud storage service provider should guarantee confidentiality, integrity, and availability (CIA) of data both in motion (while transmitting over networks) and at rest (when storing at providers disks). The issues of ”CIA” in cloud storage service have been studied in numerous works of literature including our previous studies [11] [20].

In our paper [11], we proposed a DIaaS (Data Integrity as a Service) framework, where Integrity Management Service (IMS) is available as part of cloud services. The main problems of the proposed DIaaS framework are that firstly, although we have considered the trustworthiness of Integrity Management Service and solve the problem using cryptographic approach, the assumption is that the Third Party Auditor (TPA) behaves in a responsible way, but in practice, TPA may be not so reliable as expected. Furthermore, cloud-based IMS lacks the flexibility to provide Data Integrity Service across different platforms for dynamic IoT data. Secondly, we only consider the scenario where Data Owners (DOs) need to verify their data stored in the cloud, but in practice, there has been increasing demand for sharing data with Data Consumers (DCs) who do not necessarily trust the same TPA as data owners do. In this work, we take the above concerns into consideration and suggest a blockchain based Data Integrity Service framework for IoT data.

The blockchain is the technology that lies behind Bitcoin. It enables a fully decentralized system. After the invention of the blockchain, various efforts have been made to use blockchain to decentralize the infrastructure of the current internet services [1]. In fact, there have been proposals of using blockchain for data integrity verification to benefit both blockchain system and data storage. For example, Retricoin [16] is trying to substitute the energy wasteful Proof of Work (PoW) to Proofs of Retrievability (PoR) of large size files for coin generation and data integrity verification. These approaches point to a promising future of verifying the integrity of essential data stored in a fully decentralized way. However bright the future seems to be, it will not be practical until the data storage can be decentralized with acceptable efficiency in the first place. For now, we have to consider using cloud storage service. But for data integrity service, it is worth trying to adopt a decentralized framework. In

978-1-5386-0752-7/17 $31.00 © 2017 IEEE
DOI 10.1109/ICWS.2017.54

this paper, we propose a blockchain based framework to enable decentralized data integrity verification for Internet of Things(IoT) data stored in the semi-trusted cloud. The key contributions of this paper can be summarized as follows.

- The paper replaces Integrity Management Service from the centralized node with a fully decentralized blockchain based Data Integrity Service. This eliminates the trust requirements on Third Party Auditors and increases the reliability of Data Integrity Service.
- The paper proposes protocols for data integrity verification in a fully decentralized environment and a framework suitable for both data owners and data consumers to verify specific data without relying on any single third party auditor.
- The paper demonstrates the feasibility of the proposed protocols and framework by implementing a proof-of-concept demonstrator on a private blockchain system.

The rest of the paper is organized as follows. In Section II, we review the related works with the aim of providing the motivation of our research. In Section III, background information about the related technologies is presented. In Section IV, we describe the key components enabling the proposed decentralized Data Integrity Service Framework. In Section V, we present data integrity verification protocols in a fully decentralized environment. We have implemented the protocols with smart contracts in a private Ethereum blockchain system. The prototype implementation is described in Section VI. A performance analysis of test results is presented in Section VII. We draw the conclusion and propose potential future work in the last Section VIII.

II. Related work

Data integrity is considered as one of the key security issues about data storage, which can either be cloud data storage [7] or distributed data storage [18]. In this paper, we focus only on cloud storage data integrity.

We divide previous literature about data integrity into four categories: Storage Server Technologies, Protocols, Standards [6] and Storage Architecture [17]. There are mainly two most commonly referred integrity assurance techniques falls into Storage Server Technologies, namely RAID [4] and Checksumming [13]. In protocols category, the most important ones are Provable Data Possession (PDP) [2] and Proofs of Retrievability (PoR) [8]. Both of the works were aiming to verify the integrity of large data stored in semi-trusted(PoR) or untrusted(PDP) servers without retrieving the data. By adopting these approaches, the I/O costs for data integrity verification could be reduced, which helps increase the efficiency of verifying big data. The basic idea behind the two proposals are the same: the data is separated into blocks and each block is accompanied by a small piece of metadata for verification. The main difference is I/O cost concern: PDP proposes an algorithm to verify a majority of the data blocks through verifying a small number of blocks, while PoR aims at verification of all data blocks through storage of redundantly encoded client data.

The research works from the four categories are not strictly isolated, they are often related with a different focus. For example, [17] enables public verifiability and infinite verifications based on PoR model. [5] targets to solve the problem of PDP used to verify multiple replicas of data in cloud computing model. [21] is proposed for verifying the integrity of dynamic data in multi-replica.

Our work can be categorized to Storage Architecture in a sense that we lay a foundation for previous protocols running in a decentralized environment. Our proposed framework is developed based on the principles of SOA and Web services. We aim to solve the architectural issues involved in data integrity verification.

III. Background

A. Blockchain

In general, the blockchain is a time-stamped chain of blocks jointly maintained by every participating node. *Blocks* are containers that aggregate *transactions*. The blocks are chained together cryptographically: each block is digitally signed and 'chained' to the previous block by including that block's hash value. New blocks can only be appended to the end of the chain, thus the blockchain provides an immutable data storage: existing transactions cannot be updated or deleted. This is the primary feature of the blockchain we take advantage over in this work. In other words, the occurrence of transactions can be trusted without relying on any third party authority. The immutable chain of historical transaction provides *non-repudiation* of the stored data. Cryptography and digital signatures are used to prove identity and authenticity and to enforce read and write access control to the blockchain.

The first generation of blockchains, introduced by Bitcoin, were public ledgers for monetary transactions. The second generation of blockchains provides a general, programmable infrastructure with a public ledger that records the computational results. *Smart contracts* [12] were introduced as autonomous programs that are deployed and running across the blockchain network. Smart contracts can express triggers, conditions, and even an entire business process [19]. For example, a smart contract-enabled escrow service can hold funds until the obligations defined in the smart contract have been fulfilled. Ethereum is the most popular second-generation blockchain and views a smart contract as a first-class element.

To sum up, the blockchain is immutable and deployed in a fully decentralized way with public auditability. Blockchain smart contract could implement reliable autonomous program execution.

B. Peer-to-Peer File System

There has been substantial research about building a global distributed file system. Some systems have seen successful while others failed completely. AFS is the most successful one in the academic area. In industry, Napster, KaZaA, and BitTorrent are deployed to support over 100 million simultaneous users. The recent effort is to build general file-systems that offer global, low-latency, and decentralized distribution[3]. The Peer-to-Peer (P2P) scheme is a natural successor of client-server model that was designed for small-scale distributed environments where the server has greater processing capability. Symmetric communication between the peers is the most obvious characteristic in a P2P network. Every peer is capable of being both a client and a server. P2P systems solve the bandwidth problem in sharing files from a server to clients. It enables direct communication between peers. Peers can share files with each other by pieces instead of all requesting files from a server at the same time. This substantially enhances the scalability and efficiency of file sharing.

IV. Key components in Blockchain based Date Integrity Service Framework

Our proposed framework is shown in Figure 1. There are four parts namely Data Owners Application (DOA), Data Consumer Applications (DCAs), Cloud Storage Service and Blockchain. The CSS may be further divided into private CSS and public CSS. In the rest of this paper, we assume there is one DOA which is responsible for data generation and uploading to CSS. The data integrity verification will involve multiple DOAs and DCAs. Data Integrity Service (DIS) is built upon Blockchain system. Those who need DIS should start blockchain client on their nodes first. Each node is free to join or leave the blockchain network. To simplify our proposed service framework, CSS only mean Cloud Storage Service. But in practice Cloud can also work as a blockchain node. Data Integrity Service (DIS) is implemented by smart contract living on the blockchain. The implementation is fully decentralized, which enables a higher efficiency and better reliability of DIS.

For the several concerns about the security and efficiency of blockchain system, which have a substantial influence on our proposal, we made the following assumptions. The first assumption is that 51% attack [1] is rarely possible if all the participating nodes seek to benefit themselves. The most obvious proof is Bitcoin blockchain, instead of initiating 51% attack, a malicious attacker would be more willing to use the computation power to mine. The second assumption is that blockchain consensus can be reached within a short time. Ethereum blockchain is expected to enable consensus with an average duration of 12.6s. Although at the time when we write this paper, the blockchain consensus time is still longer than this (16-18s per block), and may even fail under bad network conditions, we believe a constant short time consensus can be implemented in the near future with the progress of network and consensus algorithms.

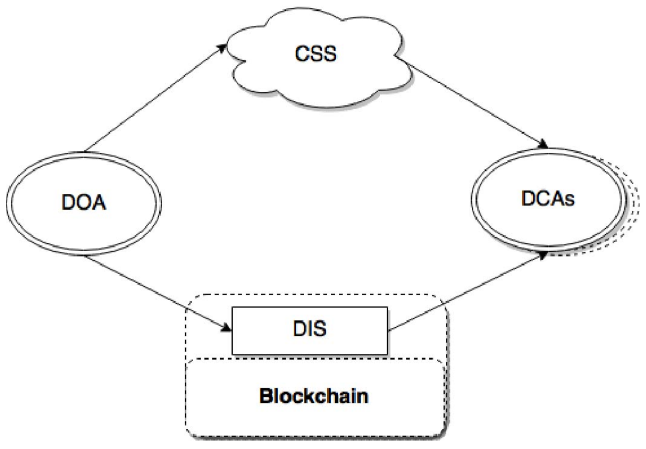

Figure 1. Illustration of Blockchain based Data Integrity Service Framework

A. Blockchain

Blockchain lays the foundation of our proposed Data Integrity Service (DIS). Both DOA and DCAs should join blockchain network at the initialization stage. A key pair would be generated, and the public key would be used as each node's account in blockchain system. Transactions can be completed if the party's account has enough *gas*. The account can only be accessed if a corresponding secret key is presented.

We assume there are enough nodes willing to be miners for keeping the blockchain system working. As a matter of fact, by adopting blockchain, our framework enables pay per transaction Data Integrity Service, *gas* is only paid when Data Owner App needs to interact with the smart contract. This significantly increases the flexibility of data integrity service when compare with cloud IMS model in our previous work [11]. It is often challenging and unnecessary for the DOAs to get *gas* by being miners because of computation power limitations. Therefore, Cloud Service Providers (CSPs) will not lose their profit in our framework, because they can act as blockchain miners to earn *gas*. Later cloud Service Providers may use the earned *gas* to trade with DOAs. DOAs would need the *gas* for the blockchain based Data Integrity Service. DOAs can use their data to trade for *gas* or financial awards with DCAs who have the flexibility to act as miners or not according to their hardware conditions and financial situations.

B. Data Integrity Service (DIS)

Data Integrity Service is implemented by smart contract. The information to be recorded in blockchain through smart contract should be encrypted locally first so that unauthorized party has no access. The account of each party is used to interact with the smart contract, all the onchain transactions with smart contract can be transparently audited. After the blockchain service is started, the blockchain data on a node will be synchronized with the whole blockchain network. DOA will need to write data into blockchain through smart contract, the data would become valid and accessible to other nodes only after the consensus of the blockchain. But if DOAs or DCAs are trying to read from blockchain through smart contract, the speed should be fast since they are actually reading data from their local synchronized data sets. This feature makes our DIS have higher efficiency than that of the cloud-based IMS proposed in our paper [11]. Upon the deployment of smart contract, participants can interact with it anytime, the Data Integrity Service cannot be terminated or modified by any participating party but the author. It would always be available as long as blockchain system works. The shutdown of blockchain system requires all the participating nodes to stop their blockchain services, which is much more complicated than terminating those services hosted by centralized providers.

In summary, compare with cloud-based IMS [11], our proposed blockchain based Data Integrity Service (DIS) has higher efficiency and more reliable.

C. *Cloud Storage Service*

All of the cloud computing service providers provide data storage. For example, Amazon S3, IBM BlueMix, Microsoft Azure and digital ocean. They all provide flexible cloud storage services according to clients' economic status and application requirements. In our framework, cloud storage service provides general purpose data storage for data owners, while the P2P file system is used for data sharing between data owners and data consumers.

V. Illustration of Decentrazlied data integrity verifcication protocols

In this section, we will focus on illustrating how our proposed decentralized data integrity verification protocols can be applied to existing Cloud Storage Service. In our framework, both blockchain and P2P file system are considered as services as addressed before. Table I shows some notations and their description referred to in this paper.

A. *System initialization*

Blockchain system initialization. Blockchain should be started on every participating node from Data Ower Apps (DOAs) and Data Consumer Apps(DCAs). Cloud can join Blockchain network and be miner if they want to profit from our service framework. Blockchain service is initialized by four steps.

- Blockchain client and its dependencies installation.
- Generate a public and private key pair before the public key is used as account to interact with Data Integrity Service Smart Contract (DISSC).
- Join the blockchain network where the Data Integrity Service Smart Contrac lives.
- Prepare *gas* either by being a miner or bought from *gas* providers. Cloud Service Providers or Data Consumer Apps may act as the *gas* providers.

2. P2PFS Service initialization. P2P File System only needs to be initialized by Cloud and Data Consumer Apps, and the initialization is as follows:

- P2PFS client and its dependencies installation.
- P2PFS repository configuration.
- P2PFS peer discovery function verification.

B. *Protocol I: Data Integrity Verification: DOA to CSS-Y*

The Protocol I is illustrated in Table II. It enables data integrity verification in the situation that Data Owner Application (DOA) needs to verify the integrity of data stored in CSS which supports cryptographic functionalities for the data integrity such as the return of a hash value of the stored data in the cloud, we use CSS-Y (Y: support cryptographic functionalities) to represent such CSS. First of all, DOA on IoT device generates different sizes of data blocks. We call one piece of the data blocks is called a Data Block. Upon the generation of each data block, DOA uses local encryption library to get the hash of the Data Block before it is uploaded to CSS. The result is presented as Hash(DataBlock). DOA on IoT device encrypt the Hash(DataBlock) locally and then write to the DISSC using a unique data Block ID, which can be used to query the encrypted Hash(DataOject) later. The reason why Hash(DataBlock) needs to be encrypted is that blockchain is a fully decentralized system where all the onchain contents can be inspected by all participating nodes. After DOA upload the Data Block to Cloud Storage Sevice (CSS), the Data Block should be stored with the same data Block ID with which the Hash of the Data Block can be retrieved. If the CSS has another Object ID for the Data Block, CSS needs to send back the Object ID and DOA makes a connection between the data Block ID and the

Table I
NOTATIONS

Notation	Description
DOAs:	Data Owner Apps
CSPs:	Cloud Service Providers
CSS:	Cloud Storage Service
DCAs:	Data Consumer Apps
P2PFS:	Peer-to-Peer File System
DIS:	Data Integrity Service
DISSC:	Data Integrity Service Smart Contract

Table II
PROTOCOL I: DATA INTEGRITY VERIFICATION: DOA TO CSS-Y

1	DOA IoT Device	:DataBlock
2	DOA IoT Device	:Hash(DataBlock)
3	DOA IoT Device	:encrypt Hash(DataBlock)
4	DOA IoT Device $\rightarrow$ DISSC	:encrypted[Hash(DataBlock)]
5	DOA IoT Device $\rightarrow$ CSS	:DataBlock
6	CSS $\rightarrow$ DOA IoT Device	:ObjectID
7	DOA IoT Device $\rightarrow$ DISSC	:ObjectID=BlockID
8	DOA $\rightarrow$ CSS	:BlockID
9	CSS $\rightarrow$ DOA	:Hash(DataBlock1)
10	DOA $\rightarrow$ DISSC	:BlockID
11	DOA $\leftarrow$ DISSC	:Encrypted[Hash(DataBlock)
12	DOA	: Decrypt(encrypted[Hash(DataBlock)])
13	DOA	:comp(Hash(DataBlock1),Hash(DataBlock))

Object ID in DISSC. We identify Object ID as the identity of Data Block stored in CSS. When DOA on IoT device or DOAs hosted by other machines than the IoT device needs to verify data set integrity, they use some object IDs of the data blocks in that data set to challenge CSS. CSS use the object ID to calculate the hash of the corresponding Data Block and return it to DOA. After receiving the feedback from CSS, DOA uses the block ID, which is connected to object ID, to query DISSC and get the encrypted Hash of the Data Block. DOA decrypt the encrypted Hash of Data Block and compare it with the one feedback from CSS.

C. Protocol II: Data Integrity Verification: DOA to CSS-N

The Protocol II is illustrated in Table III. It enables data integrity verification in the situation that Data Owner Application (DOA) needs to verify the integrity of data stored in CSS which does not support cryptographic functionalities for the data integrity such as the return of a hash value of the stored data in the cloud, we use CSS-N (N: Not support cryptographic functionalities) to represent such CSS. The data uploading process is exactly same as that in protocol I. Now that the CSS doesn't support cryptographic functionalities. Sample data blocks will need to be retrieved. DOA provide BlockID to CSS. CSS sends back the corresponding DataBlock that may be different from the one uploaded, therefore we call it DataBlock1. DOA will need to calculate the hash of DataBlock1 and compare it with the hash of the previous DataBlock written in DISSC. An optimized method for retrieving sample data blocks is to enable CSS with the P2P file system. Upon moving the Data Block to the P2P file system, CSS will get a hash of the Data Block. The hash is the unique address of the Data Block in P2P file system. CSS can feedback the hash. If the DOA uses the same library as P2P file system generating the hash, then DOA can use the hash to compare with the information in DISSC directly. Even if DOA uses a different library for hash generation, the P2P file system will speed up the file retrieving process.

D. Protocol III: Data Integrity Verification: DCAs to CSS-Y

The Protocol III is illustrated in Table IV. It enables data integrity verification in the situation that Data Consumer Application (DOA) needs to verify the integrity of data owned by DOA and stored in CSS-Y. DCAs and DOA will first reach an agreement on which dataset needs to be shared. The agreement will involve data integrity condition under which DCAs must pay DOA for the data set and the penalty for DOA if they fail to provide the claimed data integrity. The agreement is implemented by smart contract. It is called Owner Consumer Agreement Smart Contract (OCASC) and

Table III
PROTOCOL II: DATA INTEGRITY VERIFICATION: DOA TO CSS-N

1	DOA IoT Device	:DataBlock
2	DOA IoT Device	:Hash(DataBlock)
3	DOA IoT Device	:encrypt Hash(DataBlock)
4	DOA IoT Device $\rightarrow$ DISSC	:encrypted[Hash(DataBlock)]
5	DOA IoT Device $\rightarrow$ CSS	:DataBlock
6	CSS $\rightarrow$ DOA IoT Device	:ObjectID
7	DOA IoT Device $\rightarrow$ DISSC	:BlockID=ObjectID
8	DOA $\rightarrow$ CSS	:BlockID
9	CSS $\rightarrow$ DOA	:DataBlock1
10	DOA	:Hash(DataBlock1)
11	DOA $\rightarrow$ DISSC	:BlockID
12	DOA $\leftarrow$ DISSC	:Encrypted[Hash(DataBlock)
13	DOA	: Decrypt(Encrypted(Hash(DataBlock)))
14	DOA	:comp(Hash(Block1),Hash(DataBlock))
		With P2P File System
9	CSS	:DataBlock1 $\rightarrow$ P2PFS
		Same Hash Library with P2PFS
10	CSS $\rightarrow$ DOA	:P2PHash(DataBlock1)
11	DOA $\rightarrow$ DISSC	:BlockID
12	DOA $\leftarrow$ DISSC	:Encrypted[Hash(DataBlock)
13	DOA	: Decrypt(Encrypted(Hash(DataBlock)))
14	DOA	:comp(P2PHash(Block1),Hash(DataBlock))
		Different Hash Library from P2PFS
10	DOAs $\leftarrow$ CSS	:DataBlock1 from P2PFS
11	DOA	:Hash(DataBlock1)
12	DOA $\rightarrow$ DISSC	:BlockID
13	DOA $\leftarrow$ DISSC	:Encrypted[Hash(DataBlock)
14	DOA	: Decrypt(Encrypted(Hash(DataBlock)))
15	DOA	:comp(Hash(Block1),Hash(DataBlock))

Table IV
PROTOCOL III: DATA INTEGRITY VERIFICATION: DCAS TO CSS-Y

1	DOA, DCAs	:OCASC
2	DOA, CSS	:ASC
3	DOA $\rightarrow$ DCAs	:BlockIDs,Info[CSS]
4	DCAs	:Algorithem
5	DOA $\rightarrow$ DCAs	:Keys[BlockIDs]
6	DCAs $\rightarrow$ CSS	:BlockID
7	DCAs $\leftarrow$ CSS	:Hash(DataBlock1)
9	DCAs $\rightarrow$ DISSC	:BlockID
10	DCAs $\leftarrow$ DISSC	:Encrypted[Hash(DataBlock)
11	DCAs	: Decrypt(Encrypted(Hash(DataBlock)))
12	DCAs	: comp(Hash(DataBlock1),Hash(DataBlock))

is part of the blockchain based Data Integrity Service (DIS). DOA sign another smart contract called Authorizing Smart Contract (ASC) with CSS which is also part of DIS. In the contract, it is stated that the requests from any DCAs who have signed OCASC with the DOA can be provided with Hashes of Data Blocks that belongs to DOA. DOA offer the BlockIDs of the Data Blocks in the data set to be shared and CSS information to DCAs.The BlockID can be used not only to obtain hash information of Data Block from CSS but also to access the encrypted hash information of the Data Block. Thus DOA should also provide the decryption keys of those Data Blocks to be shared. DCAs then provide the BlockIDs of the Data Blocks to CSS according to a certain algorithm based on PDP or PoR. CSS feedbacks the corresponding hashes of the Data Blocks. DCAs compare the hashes of the Data Blocks with the decrypted hashes of the Data Blocks to decide data integrity.

Table V
PROTOCOL IV: DATA INTEGRITY VERIFICATION: DCAS TO CSS-N

1	DOA, DCAs	:OCASC
2	DOA, CSS	:ASC
3	DCAs	:Algorithem
4	DOA $\rightarrow$ DCAs	:BlockIDs,Keys[BlockIDs],Info[CSS]
5	DCAs $\rightarrow$ CSS	:BlockID
6	DCAs $\leftarrow$ CSS	:DataBlock1
7	DCAs	:Hash(DataBlock1)
8	DCAs $\rightarrow$ DISSC	:BlockID
9	DCAs $\leftarrow$ DISSC	:Encrypted[Hash(DataBlock)
10	DCAs	: Decrypt(Encrypted(Hash(DataBlock)))
11	DCAs	: comp(Hash(DataBlock1),Hash(DataBlock))

E. Protocol IV: Data Integrity Verification: DCAs to CSS-N

The Protocol IV is illustrated in Table V. It enables data integrity verification in the situation that Data Consumer Application(DOA) needs to verify the integrity of data owned by DOA and stored in CSS-N. The procedures before DOA sign Authorizing Smart Contract (ASC) with CSS are exactly same as Protocol III. But in this protocol, in the ASC, it should be stated that the requests from any DCAs who have signed OCASC with the DOA can be provided with Data Blocks that belongs to DOA for a limited number of times. Accessing the limited number of the Data Blocks in the data set to be shared will not reveal any useful information about the whole data set. DCAs use a certain algorithm to decide which should be the limited Data Blocks to be retrieved from the data set to be shared for integrity verification. Upon DCAs decided which Data Blocks are to be verified, DOAs offer the BlockIDs of the Data Blocks, decryption keys of the onchain information of those Data Blocks and necessary CSS information to DCAs. Let's consider the verification of one of the Data Blocks because the verification of the rest is the same. DCAs retrieve Data Block from CSS called DataBlock1 because the integrity may have been corrupted. The Hash of the DataBlock1 is calculated first. Then DCAs refer to DISSC for the information of the same Data Block, if they are the same then Data Integrity is verified. The procedures are very similar to what has been stated in Protocol II and we will not address again.

VI. SYSTEM IMPLEMENTATION

The detailed implementation of the prototype system is stated in this section.
Figure 2 shows a detailed structure of the proposed service framework, our prototype is implemented based on this.

Data owner Applications (DOAs) are running on both Personal Computers (PCs) and IoT devices. DOAs on IoT devices are responsible for data generation, uploading to CSS, hash generation, encrypt the generated hash and writing to DISSC. DOAs on PCs can be used to verify data

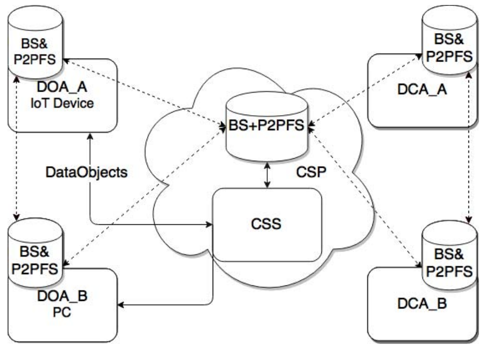

Figure 2. Prototype Implementation

integrity. DCAs can run on PCs and clouds. They all should have capability of running Blockchain Service (BS) and P2PFS. Modern PCs are capable of running blockchain without a doubt. IoT devices referred to here is Single Board Computers that have General Purpose Inputs/Outputs (GPIOs) to collect sensor data. Most of the present Single Board Computers such as Raspberry Pi have such capability. Data blocks from sensors can be processed directly by IoT devices before uploading to CSS. The blockchain system is implemented by Ethereum since it is the most mature blockchain system that supports smart contract. The distributed P2P File System is implemented by IPFS, which stands for Inter-Planetary File system, it is an attempt to share files in an HTTP manner, we use IPFS to facilitate data verification in Protocol II and Protocol IV.

VII. Performance analysis

A prototype system has been developed to evaluate the feasibility of our framework and the relevant protocols. Since IoT data is dynamic in nature, we assume that the data is available in the form of different sizes of data blocks and a data set is composed of some those data blocks. In general, we should check the integrity of data blocks before the integrity of the whole data sets can be verified. Our prototype system is built upon a private blockchain with a minimum of 4 nodes at the moment. One node is hosting CSS, the blockchain, and P2P file system. A personal computer is running both DOA and DCA. An IoT device is running DOA for data block generation, uploading to CSS, etc. A public cloud is running DCA to test the data block download efficiency under different network conditions.The test environment setting is listed below:

- Raspberry Pi 2: Run DOA.
- Server1: Work as private Cloud Service Provider and blockchain miner.
- CloudPub (Public): Run DCA.
- PC_1 : Run both DOA and DCA.

Figure 3 shows the efficiency to verify Data Integrity from DOA to CSS-Y. In Figure 4, the comparison results show that the time used by PC_1 is substantially shorter than Pi2. At least two factors influence the results, the first one is the computation capability, and the second one is the client version of Ethereum. Figure 5 shows the time spent by various DOAs and DCAs for querying DISSC. Figure 6 shows the time spent for retrieving different sizes of data blocks in CSS-N model. The test results show that our proposed framework can support the integrity verification of data blocks by multiple DOAs or DCAs.

VIII. Conclusion and future work

There are three key aspects of cloud data security: confidentiality, integrity, and availability (also known as CIA). In our previous work, we introduced a concept of Data Integrity as a Service to address the data integrity concerns in the cloud storage service. The major shortcoming of the existing techniques is that the data owner has to rely on a trusted Third Party Authority to complete the data integrity verification task. However, this is a strong

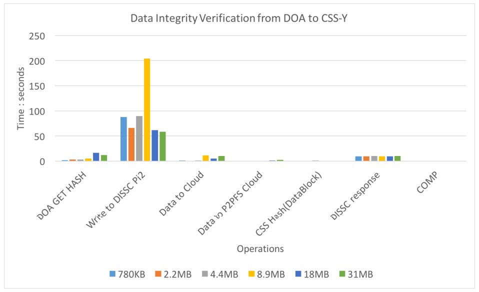

Figure 3. Performance of Data Integrity Verification from DOA to CSS-Y

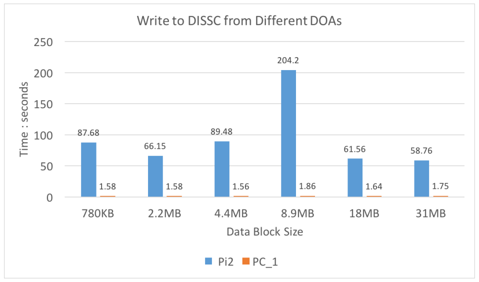

Figure 4. Comparison of writing to DISSC from different DOAs

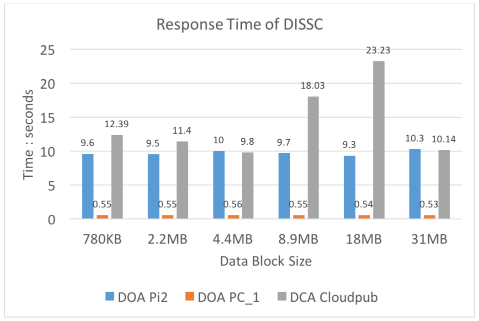

Figure 5. Response Time of DISSC to DOAs and DCAs

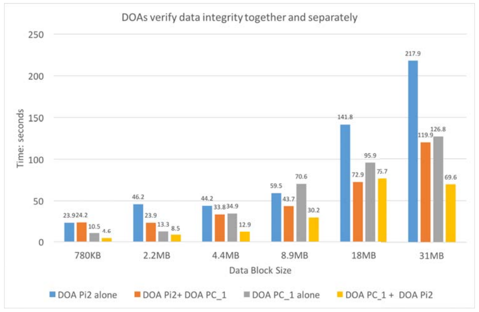

Figure 6. Time Spent for Retrieving Data Blocks

assumption that sometimes not hold. As a result, the data integrity verification results may not be trustworthy. In this paper, efforts are made to implement a blockchain based Data Integrity Service. Compare with previous works, The service framework we propose has the following advantages:

- It is more reliable. No single party cloud terminate it.
- Data Integrity Verification efficiency can be enhanced with increasing number of clients.
- It supports trading data with data consumers, and imeplement pay per transaction Data Integrity Service.

However, we only implemented the fundamental function of our proposed protocols.IoT device in our test still has low efficiency in writing to smart contract.The test environment is still of small scale. Therefore, our next step is to extend our work to:
1. Implement the algorithms in IV for ensuring data integrity of whole data set by retrieving only a portion of data blocks.
2. Implement the other blockchain smart contract based services for universal trusted data trading.

REFERENCES

[1] M. Ali, J. C. Nelson, R. Shea, and M. J. Freedman. Blockstack: A global naming and storage system secured by blockchains. In *2016 USENIX Annual Technical Conference, USENIX ATC 2016, Denver, CO, USA, June 22-24, 2016.*, pages 181–194, 2016.

[2] G. Ateniese, R. C. Burns, R. Curtmola, J. Herring, L. Kissner, Z. N. J. Peterson, and D. X. Song. Provable data possession at untrusted stores. In *Proceedings of the 2007 ACM Conference on Computer and Communications Security, CCS 2007, Alexandria, Virginia, USA, October 28-31, 2007*, pages 598–609, 2007.

[3] J. Benet. IPFS - content addressed, versioned, P2P file system. *CoRR*, abs/1407.3561, 2014.

[4] P. M. Chen, E. K. Lee, G. A. Gibson, R. H. Katz, and D. A. Patterson. RAID: high-performance, reliable secondary storage. *ACM Comput. Surv.*, 26(2):145–185, 1994.

[5] R. Curtmola, O. Khan, R. C. Burns, and G. Ateniese. MR-PDP: multiple-replica provable data possession. In *28th IEEE International Conference on Distributed Computing Systems (ICDCS 2008), 17-20 June 2008, Beijing, China*, pages 411–420, 2008.

[6] J. Gray. The transaction concept: Virtues and limitations (invited paper). In *Very Large Data Bases, 7th International Conference, September 9-11, 1981, Cannes, France, Proceedings*, pages 144–154, 1981.

[7] T. Jiang, X. Chen, and J. Ma. Public integrity auditing for shared dynamic cloud data with group user revocation. *IEEE Trans. Computers*, 65(8):2363–2373, 2016.

[8] A. Juels and B. S. K. Jr. Pors: proofs of retrievability for large files. In *Proceedings of the 2007 ACM Conference on Computer and Communications Security, CCS 2007, Alexandria, Virginia, USA, October 28-31, 2007*, pages 584–597, 2007.

[9] J. J. Jung. Computational collective intelligence with big data: Challenges and opportunities. *Future Generation Comp. Syst.*, 66:87–88, 2017.

[10] H. S. Narman, M. S. Hossain, M. Atiquzzaman, and H. Shen. Scheduling internet of things applications in cloud computing. *Annales des Télécommunications*, 72(1-2):79–93, 2017.

[11] S. Nepal, S. Chen, J. Yao, and D. Thilakanathan. Diaas: Data integrity as a service in the cloud. In *IEEE International Conference on Cloud Computing, CLOUD 2011, Washington, DC, USA, 4-9 July, 2011*, pages 308–315, 2011.

[12] S. Omohundro. Cryptocurrencies, smart contracts, and artificial intelligence. *AI Matters*, 1(2), Dec. 2014.

[13] S. Patil, A. Kashyap, G. Sivathanu, and E. Zadok. I^3fs: An in-kernel integrity checker and intrusion detection file system. In *Proceedings of the 18th Conference on Systems Administration (LISA 2004), Atlanta, USA, November 14-19, 2004*, pages 67–78, 2004.

[14] S. Peng, G. Wang, and D. Xie. Social influence analysis in social networking big data: Opportunities and challenges. *IEEE Network*, 31(1):11–17, 2017.

[15] M. M. Rathore, A. Paul, A. Ahmad, and G. Jeon. Iot-based big data: From smart city towards next generation super city planning. *Int. J. Semantic Web Inf. Syst.*, 13(1):28–47, 2017.

[16] B. Sengupta, S. Bag, S. Ruj, and K. Sakurai. Retricoin: Bitcoin based on compact proofs of retrievability. In *Proceedings of the 17th International Conference on Distributed Computing and Networking, Singapore, January 4-7, 2016*, pages 14:1–14:10, 2016.

[17] H. Shacham and B. Waters. Compact proofs of retrievability. In *Advances in Cryptology - ASIACRYPT 2008, 14th International Conference on the Theory and Application of Cryptology and Information Security, Melbourne, Australia, December 7-11, 2008. Proceedings*, pages 90–107, 2008.

[18] J. B. J. P. Shawn Wilkinson, Tome Boshevski. Storj: A peer-to-peer cloud storage network. 2016.

[19] I. Weber, X. Xu, R. Riveret, G. Governatori, A. Ponomarev, and J. Mendling. Untrusted business process monitoring and execution using blockchain. In *Intl. Conf. Business Process Management*, Sept. 2016.

[20] J. Yao, S. Chen, S. Nepal, D. Levy, and J. Zic. Truststore: Making amazon S3 trustworthy with services composition. In *10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, CCGrid 2010, 17-20 May 2010, Melbourne, Victoria, Australia*, pages 600–605, 2010.

[21] Y. Zhang, J. Ni, X. Tao, Y. Wang, and Y. Yu. Provable multiple replication data possession with full dynamics for secure cloud storage. *Concurrency and Computation: Practice and Experience*, 28(4):1161–1173, 2016.