2022 IEEE International Conference on Edge Computing and Communications (EDGE) | 978-1-6654-8140-3/22/$31.00 ©2022 IEEE | DOI: 10.1109/EDGE55608.2022.00025

Bring Trust to Edge: Secure and Decentralized IoT Framework with BFT and Permissioned Blockchain

Yusen Wu*‡, Jinghui Liao†, Phuong Nguyen*, Weisong Shi†, Yelena Yesha*
*University of Miami †Wayne State University ‡University of Maryland, Baltimore County
{ywu5, phuong3}@umbc.edu, {jinghui, weisong}@wayne.edu, yxy806@miami.edu

***Abstract*—While our society accelerates its transition to the Internet of Things, billions of IoT devices are now linked to the network. While these gadgets provide enormous convenience, they generate a large amount of data that has already beyond the network's capacity. To make matters worse, the data acquired by sensors on such IoT devices also include sensitive user data that must be appropriately treated. At the moment, the answer is to provide hub services for data storage in data centers. However, when data is housed in a centralized data center, data owners lose control of the data, since data centers are centralized solutions that rely on data owners' faith in the service provider. In addition, edge computing enables edge devices to collect, analyze, and act closer to the data source, the challenge of data privacy near the edge is also a tough nut to crack.**

A large number of user information leakage both for IoT hub and edge made the system untrusted all along. Accordingly, building a decentralized IoT system near the edge and bringing real *trust* to the edge is indispensable and significant. To eliminate the need for a centralized data hub, we present a prototype of a unique, secure, and decentralized IoT framework called Reja, which is built on a permissioned Blockchain and an intrusion-tolerant messaging system ChiosEdge, and the critical components of ChiosEdge are reliable broadcast and BFT consensus. We evaluated the latency and throughput of Reja and its sub-module ChiosEdge.

***Index Terms*—Permissioned Blockchain, Edge Computing, IoT, Data Privacy, Byzantine Fault Tolerance**

I. Introduction

The fast expansion of the Internet of Things (IoT) has resulted in the deployment of billions of intelligent IoT devices. The sensors on these gadgets continue to generate and gather useful and lucrative data for industrial companies. Industry companies now utilize IoT platforms[1] to collect data, get insight from it, and make data-driven choices. However, as more IoT devices are deployed in our everyday lives, the data acquired by such devices often include information we do not like to share with those corporations. Since the present IoT platform solutions send such privacy data to the data center as well, however, the corporations who operate the data center and IoT platform have complete access to the data. Data consumers who must create an account or get permission may access and analyze the data through a Web or an App supported by an IoT platform. Another problem with the present IoT platform solution is that the predicted proliferation of hundreds of billions of devices puts us on the verge of a transition that will reshape the electronics industry and many other sectors. The reality is, can we really trust the data? Indeed, where does the data originate, which device and when does it generate, and should we be making choices and transacting based on unvalidated data?

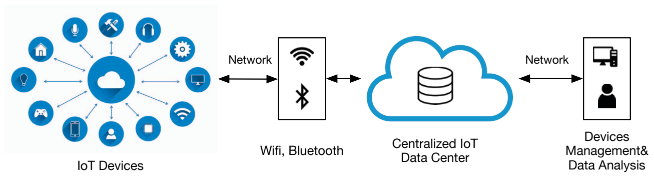
Network
Network
IoT Devices
Wifi, Bluetooth
Centralized IoT Data Center
Devices Management& Data Analysis

Figure 1. Centralized IoT Hub Platform

Meanwhile, the rise of edge computing [45] enables smart edge devices[2], such as smart routers or routing switches, to collect, analyze, and act closer to the source, lowering the communication overhead between the IoT hub and IoT devices and correspondingly lowering latency and bandwidth usage. This implies that the IoT platform and edge computing must be tightly coupled. However, here is a question: should we *trust* the outcomes of data processing on an edge device? As previously indicated, data generated by IoT devices may be inaccurate and even hazardous.

Additionally, we discovered that practically all commercial IoT solutions are centralized, which indicates that the IoT hub would gather and store all IoT data from smart devices in a data center, as seen in Figure. 1. However, sensitive data, such as patient data, are not desired to be made public or shared with a third-party data center. As a result, enhancing data credibility (e.g., establishing the trust), developing a privacy-preserving IoT platform for edge devices, and effectively integrating with edge computing to decrease communication overhead are key challenges.

Numerous research have been undertaken on IoT platform privacy issues, and academics have also recommended effective approaches to address vulnerabilities discovered in these edge IoT devices and IoT hubs [8], [9], [12], [17], [24], [31], [33], [36]–[38], [47], [49], [50], nonetheless, such works are subject to the following restrictions:

[1]Service that consists of a collection of components for data collection, sensor management, and visualization in Internet of Things projects, such as Google Cloud IoT Core, IBM Watson IoT Platform, and AWS IoT Core.

[2]IoT devices and edge devices are different in this paper. IoT devices include sensors, mobile phones, etc. Edge devices include Gateway, Setup Box, or Base Stations, etc.

(1) Insufficient knowledge about data ownership. In terms of IoT privacy, a major case study occurred in 2019 with the data loss at Wyze [7], a producer of smart IoT devices. Wyze revealed that for weeks, sensitive data obtained from millions of individuals were left accessible on the internet, including email addresses and health information [7]. As of 2021, the mainstream media in the United States reported on several IoT devices and IoT corporations exposing customers' personal data [4], [5].

We concur that data collected by IoT devices and kept in a personal data cloud or IoT data center may be treated as a property *asset* or *item* under property law. Why would data owners allow a third-party platform to have such easy access to all of their sensitive data, including their home addresses, phone numbers, bank accounts, and even healthcare records?

(2) Availability, reliability, and single point of failure (SPOF). The possibility of a system being operational at a particular moment is referred to as availability. Numerous sorts of assaults, such as the iconic Mirai botnet and its variants, are exploding in popularity as their new tactics and strategies evolve. Apart from disrupting a large portion of the global network, these types of cyber-attacks have resulted in significant harm to national property. Kaspersky [6] said that its honeypots identified more than 100 million assaults from 276,000 distinct IP addresses in the first half of 2020, about nine times the amount of attacks in 2018. If a device fails or is subjected to a DDoS assault, it will cease delivering data to the IoT platform, or, in the worst-case scenario, the whole IoT platform would fail, jeopardizing data availability. As a result, removing SPOF lays the groundwork for high availability. Another constraint on developing a secure IoT system is its resiliency in the face of Byzantine (arbitrary) failures. We discovered that state machine replication and reliable broadcasting protocols, such as BFT-SMaRt [10], were used to remove SPOF, although operating a Byzantine Fault Tolerance (BFT) consensus protocol across thousands of IoT devices is difficult due to the communication cost. As a result, various lightweight consensus algorithms have been suggested that can be implemented on smart IoT devices, but they still need a black-box trust execution environment (TEE) [13] to reduce $3f + 1$ to $2f + 1$ which is also impractical for running BFT consensus among IoT devices.

(3) Decentralized BFT-based edge storage system is required. Some decentralized database have been proposed, for example, Gheorghe et al. [30] and Cui et al. [19] provide a decentralized storage for data generated by different edge devices, but rare of them use BFT core as the underlying engine to tolerate Byzantine faults and bring real *trust*. A *trust* here means the data consumer fully trust where the data comes from and the current data that you received never be compromised in the database by adversaries.

(4) Limited confidentiality and access control. Another critical security challenge in IoT is data confidentiality, specifically how to prevent unauthorized users from accessing IoT devices and resources (data, apps, and services). Confidentiality in the IoT and edge must be tightly coupled with access control. Access control options now available in the IoT include discretionary access control (DAC) [44], mandatory access control (MAC) [39], role-based access control (RBAC) [28], organization-based access control (OBAC) [35], attribute-based access control (ABAC) [48], and usage control (UC) [40].

The downsides of contemporary access control in IoT include the following: 1) some hacking and security difficulties exist; 2) the technology is still in its infancy; and 3) the system is complicated and expensive. Taking advantage of fabric blockchain, we offer resource-based access control via fabric blockchain. The fabric uses access control lists (ACLs) to manage access to resources by associating a policy with a resource, and the chaincode (smart contracts) can utilize the client certificate for access control decisions [3].

Keeping these concerns and constraints in mind, we offer Reja, a unique, secure, and decentralized IoT framework built on a permissioned Blockchain and an intrusion-tolerant messaging system ChiosEdge with reliable broadcast and Byzantine fault tolerance (BFT) consensus running on edge devices as fundamental components. Because all active BFT nodes retain complete copies of the data, a reliable broadcast and BFT consensus are deemed dependable. Thus, even if one node goes down, the sensitive data remains accessible to all other network members.

CONTRIBUTIONS. In summary, our contributions are as follows:

- We use permissioned Blockchain and BFT as core underlying modules to instill true confidence in the data that is gathered. The objective is to imbue each gadget with an identity at the point of manufacture and maintain it throughout its existence in an immutable ledger. Additionally, a gadget with an identity may build a history tie that blockchain owners can trace or follow. The primary contribution is to identify data ownership and to get actionable information at the appropriate moment.
- We implemented ChiosEdge, a BFT storage system, to install on several edge devices in order to replicate encrypted sensitive data in personal cloud databases or local databases.
- An adaptive threshold signature (ATS) is proposed for decentralized data governance. A user who wants to access the edge data stored in the secure cloud or local databases needs digital consent and a private key. Multiple investigators are offered here to vote for the access permission.
- Finally, we evaluated the latency and throughput of Reja and its sub-module ChiosEdge.

Organizations. The Section II discusses similar work. Section III discusses the background, which include open issues in edge computing, permissioned Blockchain, data consent, smart contracts and chaincode, and Byzantine Fault Tolerance. Section IV provides an overview of the framework. Section

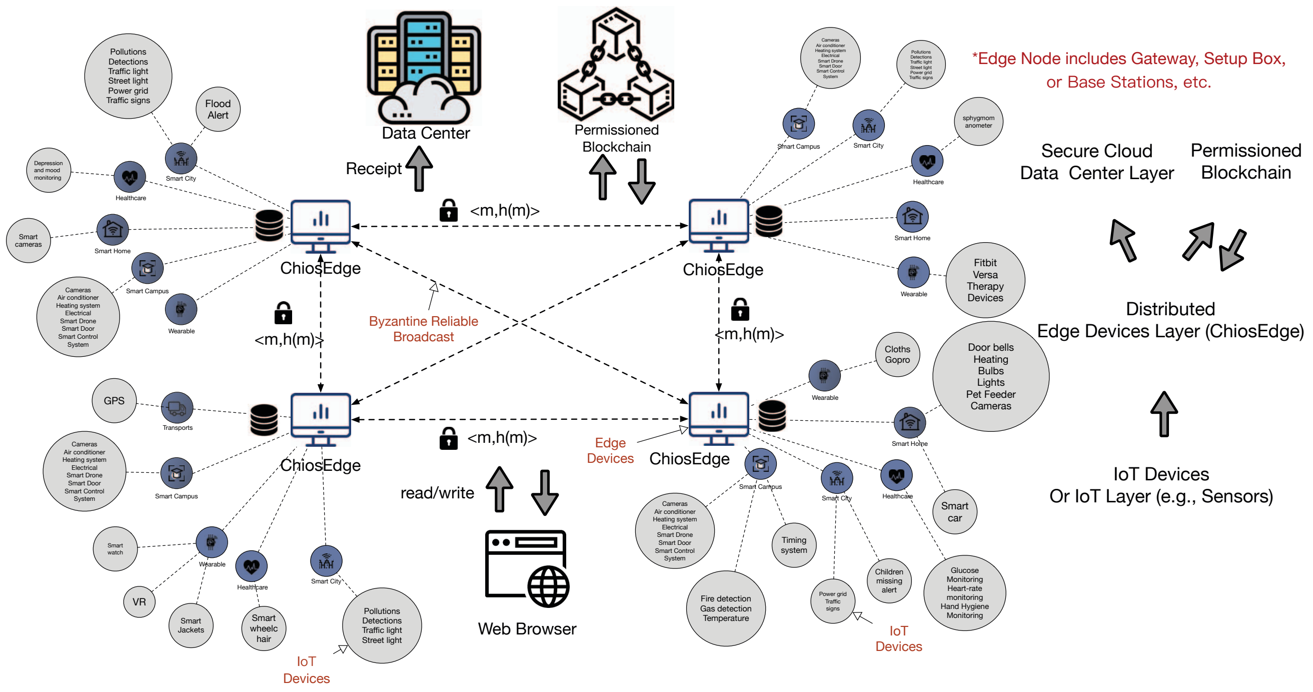
Data Center
Permissioned Blockchain
Receipt
<m,h(m)>
*Edge Node includes Gateway, Setup Box, or Base Stations, etc.
Secure Cloud Data Center Layer
Permissioned Blockchain
Distributed Edge Devices Layer (ChiosEdge)
IoT Devices Or IoT Layer (e.g., Sensors)
ChiosEdge
ChiosEdge
ChiosEdge
ChiosEdge
Byzantine Reliable Broadcast
Edge Devices
IoT Devices
IoT Devices
read/write
Web Browser
Pollutions Detections Traffic light Street light Power grid Traffic signs
Flood Alert
Depression and mood monitoring
Smart City
Healthcare
Smart cameras
Smart Home
Smart Campus
Wearable
Cameras Air conditioner Heating system Electrical Smart Drone Smart Door Smart Control System
GPS
Transports
Smart watch
VR
Smart Jackets
Smart wheelc hair
Pollutions Detections Traffic light Street light
sphygmom anometer
Fitbit Versa Therapy Devices
Cloths Gopro
Door bells Heating Bulbs Lights Pet Feeder Cameras
Smart car
Timing system
Fire detection Gas detection Temperature
Power grid Traffic signs
Children missing alert
Glucose Monitoring Heart-rate monitoring Hand Hygiene Monitoring

Figure 2. Reja Architecture and Workflow with BFT consensus and Reliable Broadcast. A detailed write and read workflow is discussed in Section V-A

V introduces the system's detailed components and structure. Section VI discusses the implementations, Section VII evaluates Reja, and Section IX concludes.

II. Related Work

Blockchain technology has the potential to solve significant obstacles in IoT applications, such as SPOF and a lack of trust. Some publications discussed a hybrid Blockchain solution for data security and user privacy, while others discussed Blockchain for edge computing. Finally, some academics offered lightweight BFT protocols for data security to replace PoW.

Ali Dorri et al. [25] proposed using lightweight hashing to detect any changes in transactions and eliminates the proof of work (POW) and the concept of coins in a public Blockchain network. AnaReyna et al. [41] proposed a hybrid design where only part of the interactions and data take place in the Blockchain and the rest are directly shared between the IoT devices with fog computing. AnaReyna et al. also proposed a hybrid design that only handle part of the interactions and data on the Blockchain while processing the rest in the edge computing devices [41]. Kazi et al. [42] proposed an idea of distributed intelligence that can perform instant decision-making and reduce unnecessary data transferring to the cloud, which addressed various security challenges in the IoT paradigm, however, this paper is only a high-level idea without any specific solutions for reducing the irrelevant data transfer. Yu et al. [15] proposed a hybrid Blockchain-based privacy-preserving electronic medical records sharing scheme across medical information control system, where the sensitive data is stored in a permissioned Blockchain and other data is stored in a public Blockchain. Saide et al. [52] proposed a hybrid Blockchain, named zkCrowd, that integrates with a hybrid Blockchain structure, smart contract, dual ledgers, and dual consensus protocols to secure communications, verify transactions, and preserve privacy. By utilizing DPOS and PBFT consensus protocols, the transaction verification efficiency is significantly increased and the energy consumption and transaction latency is reduced in the crowdsourcing system. [20] is also a hybrid Blockchain model proposed by Cui. [20] consists of two parts: local Blockchain and public Blockchain. To preserve the IoT users' privacy and avoid information leakage to the main Blockchain, an interconnection position, called bridge, is introduced by Firoozjaei et al. [21] to isolate IoT users' peer-to-peer transactions and link the main Blockchain to its subnetwork Blockchain(s) (e.g., one main Blockchain, and others are subnetwork Blockchain for sensitive data storage) in a hybrid model. Fan et al. [27] proposed a hybrid Blockchain for federated learning in edge computing, and both public Blockchain and permissioned Blockchain are involved. They design and implement a smart contract in public Blockchain to facilitate an automatic, autonomous, and auditable rational reverse auction mechanism among edge nodes and leverage the payment channel technique in public Blockchain to enable credible, fast, low-cost, and high-frequency payment transactions between requesters and edge nodes. Desai et al. [22] also proposed a novel hybrid Blockchain architecture that combines private and public Blockchains to allow sensitive bids to be opened on a private Blockchain so such that only the auctioneer can learn the bids. He aslo proposed BlockFLA [23] to build an accountable federated learning system via public Blockchain and permissioned Blockchain. Zhu et al. [51] proposed a design for privacy-preserving crowdsourcing platforms also using both public and permissioned Blockchain. Other papers, [11], [43], [53] also mentioned public and private Blockchain related architecture for enhancing the data security and user privacy.

Even though a hybrid design can reduce the storage overhead in a Blockchain, its security issues in the edge devices still remain a serious concern, for example, some vulnerable fog devices might be crashed or attacked by malicious adversaries. Most hybird Blockchain IoT or edge computing systems are still under SPOF attack and with limited availability.

III. Background

A. Byzantine Fault Tolerance

Byzantine Fault Tolerance (BFT) [16] is a computer system's ability to continue operating even if some of its nodes fail or act maliciously. In this paper, we use ChiosEdge as the BFT system, and BFT-SMaRt [10] as the underlying BFT core in ChiosEdge. BFT-SMaRt can tolerate both Byzantine faults ($3f + 1$) and crash recovery failure ($2f + 1$), for example, under $3f + 1$ consensus protocol, 4 nodes can tolerate 1 faulty node.

B. Open Challenges in Edge Computing

The edge computing use case landscape in which the early deployments have been highly customized is broad and still immature. Infrastructure and operations developers will need to develop a multiyear edge computing strategy that addresses the challenges of diversity, location, protection, and data privacy. Our Reja and its sub-module system ChiosEdge offer a Byzantine fault tolerance consensus protocol to tolerate arbitrary faults and bring *trust* to the data itself, which concentrates on the challenges of data privacy, data protection, and data availability near the edge.

C. Chaincode and Smart Contracts

Chaincode, also known as smart contracts in Fabric [14], is a sub-module that enables the reading and updating of data on a Blockchain ledger. Chaincode specifies the transaction logic that governs the lifespan of a business entity and converts it to a Blockchain-based executable program. It establishes the manner in which business logic is packaged for deployment on a Blockchain network.

D. Data Consent

Consent [18] is an unambiguous indication of a data subject's wishes that signifies an agreement by the data owner to the processing of personal data relating to data consumers whereby that consent needs to be given in clearly defined ways. Blockchain-based digital consent has been proposed by lots of researchers and industrial companies. Taking the advantages of Blockchain, we stored digital consent in an immutable ledger on AWS Managed Blockchain. We also involve *investigators* as overseers for data governance. The *investigators* can sign signatures for offering final permission in a decentralized solution. Combining Blockchain, data consent, and threshold signature, a trusted and efficient access control platform is developed in this paper.

E. Permissioned Blockchain

A permissioned Blockchain [32], also known as a private Blockchain, is a distributed ledger that is not publicly accessible. Only certain identifiable participants can access the ledger. If a new member wishes to join, it needs to be invited and the majority of the existing on-chain members must vote *Yes* on the proposal. Compared to public Blockchain, permissioned Blockchain eliminates the Proof of Work (PoW) [29] consensus protocol in favor of Byzantine Fault Tolerance (BFT) mechanisms. It decreases system processing time and avoids sensitive data being copied to too many replicas or nodes (e.g., a public Blockchain has thousands of nodes).

IV. A Modular System Overview

Reja framework provides a modular framework allowing trade-offs between functionality, security, and efficiency. There are four major modules of Reja: a reliable BFT system ChiosEdge, secure data collection, data governance, and a permissioned Blockchain network with a customized smart contract for data consent. In addition, the underlying modules of ChiosEdge include a BFT consensus core, data read and write, and a light-weight ChiosLite running in cheap IoT devices. We simply introduce all the modules in this section and give more detailed explanations in the next section.

A. ChiosEdge Architecture and Message Flow

ChiosEdge, BFT as underlying consensus protocol, is a permissioned BFT system, which supports Byzantine reliable broadcast (brb) with a total order broadcast and cryptographical primitives (e.g., encryption and authentications). We have improved and trimmed existed functions to make them more suitable for running on the edge devices. ChiosEdge can send write/read request from client through write/read request handler threads shown in Fig. 3. All the requests need to make a consensus before proceeding. For example, a write request will broadcast message m and hashcode $h(m)$ to every replicas, and each of replicas start to verify message m and deliver m after the master node received $f + 1$ of same values. Each replica would store message m in its local database with the same timestamp and sequence order after running consensus correctly. ChiosEdge also supports IoT device registration, the IoT devices need to be registered before sending data to edge devices because ChiosEdge only accepts authorized IoT devices through the message authentication code (MAC) function. A lightweight ChiosLite is proposed to deploy on cheap IoT devices such as Raspberry Pi for securely collecting sensor data.

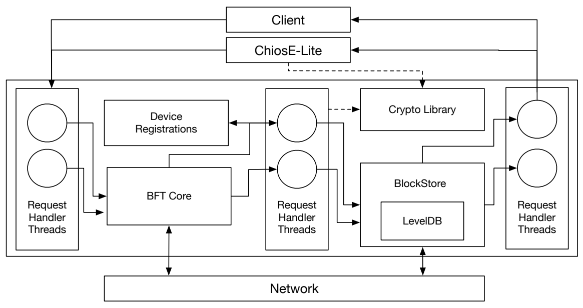

Figure 3. ChiosEdge architecture and message flow.

B. Data Consent Chaincode

Chaincode, also referred to as smart contracts in Hyperledger Fabric Blockchain (HFB), is a sub-module that you can use to read and update data on a Blockchain ledger. A data consent ledger in a Blockchain network signifies agreements by data owners to the processing of personal data relating to themselves. A data consent includes start time, end time, data owner signatures, etc.

C. Secure Data Collection

We implemented a lightweight ChiosLite that can operate on a low-cost operating system such as the Raspberry Pi. To authenticate the identity of an IoT device, an ECDSA [34] library is installed. And ChiosEdge servers gather only transactions sent by authorized devices. For sensitive data, ChiosLite may additionally encrypt transactions on the IoT devices side for sensitive data. Through the use of the private key k, the ChiosEdge servers may decode the *ciphertext* $E(m)$.

D. Reliable Broadcast for Data Governance

We describe a module named adaptive threshold signature (ATS) with reliable broadcast for data governance. A user who wants to get consent from Blockchain needs permission. A (t, n) ATS scheme is given, it consists of the following algorithms (Gen, Sga, Vrf, Com, Sva), n refers to the number of investigators, and t refers to the thresholds.

- **Key Generation Algorithm** Gen contains n nodes. Each node n_i inputs common parameters, including a security parameter l, a total number of servers n, and a threshold number t. It outputs (pk, vk, sk), where the pk is a public key, vk is a verification key, and sk = (sk_1,...,sk_n) is a list of private keys. Both pk and vk are public, and each node n_i obtains its own sk_i.
- **Shares Generation Algorithm** Sga inputs pk, m, sk_i, and outputs a share signature sig_i.
- **Share Verification Algorithm** Vrf inputs vk, m, a share signature sig_i, and outputs a single bit *True/False*.
- **Combining Algorithm** Com inputs vk , m, a set of t valid share signatures, and outputs a value Sig.
- **Signature Verification Algorithm** Sva takes a combined signature Sig as an input, m, a public key pk, and outputs *True/False*.

All the investigators will digitally generate a value sig_i (share), the ATS Com combines all the shares and outputs a value Sig, the *signature verification algorithm* Sva then verifies the signature Sig and outputs *True/False*. A write transaction can be committed and stored in the Blockchain ledger if it has been signed and verified correctly with ATS.

V. The Reja System

Reja currently supports four main modules, including ChiosEdge BFT system, ChiosLite, data consent, and decentralized governance with (t, n) adaptive threshold signature. The ChiosEdge module also includes device registration, BFT core, broadcast total order and access control. We describe the detailed functions of these four modules as follows.

A. Reja *Architecture and Workflow*

The Reja architecture includes three layers: 1), IoT devices layer (L1), 2), distributed edge devices (replicas) layer (L2), and 3), secure cloud data center and permissioned Blockchain (L3), as shown in Fig 2.

An organization, Org1, manually deploys different types of sensors or IoT devices in its local environment (e.g., patient medical wearable devices), and L2 securely collects these IoT data and commits them to the decentralized storage, ChiosEdge stands the role for tolerating Byzantine faults and reliable decentralized database. The IoT data is stored in the edge devices locally or secure cloud database.

- *For* write: Assuming one message m comes from an IoT device, first, the ChiosEdge verifies IoT identity and decrypts the *ciphertext* if message m is encrypted. Second, after executing a write consensus operation, all the edge nodes will store a copy of this message m in their local database and primary edge replica will send a log to data center as a *receipt*.
- *For* read: If other organization, for example Org2, wants to access the edge data, he needs send a read request to the investigators ask for permission, if majority of the investigators vote for *YES*, he will get the consent and private key from the Blockchain ledger, and meanwhile, the smart contract will send an endpoint link for access the data.

B. ChiosEdge *BFT System*

1) Device Registrations: As ChiosEdge only stores the data from authorized IoT devices, each device needs to be registered and verified for secure data collection. In ChiosEdge, register device identity using IP and port number, and the device id is assigned by the ChiosEdge servers. The registered device id is stored in a private config file with a unique timestamp. In addition, as the device registration also needs to run a consensus, the primary node broadcasts a registration request to every peer node (edge devices) for voting, after a *Quorum* of nodes accept this request, all the peer nodes keep a copy of this registration log (evidence) in their local databases and configuration files.

When an IoT device sends a message m to the ChiosEdge node, ChiosEdge node verifies the device id through message authentication code (MAC) and commits m if the device id has been registered.

2) ChiosEdge *with consensus BFT core:* We use BFT-SMaRt protocol as the underlying BFT core for consensus. BFT-SMaRt supports both Byzantine fault tolerance (BFT) and crash-recovery fault tolerance (CFT). As BFT includes all types of arbitrary faults and CFT only tolerates crash faults, BFT needs at least 4 replicas for tolerating 1 faulty node. In addition, BFT-SMaRt includes state transfer protocols to recover replicas from failures, *DurabilityCoordinator* and *DefaultRecoverable*. For example, *DurabilityCoordinator* stores its logs to disk and executes in parallel to mitigate latency.

3) ChiosEdge *with total order broadcast for reliability:* Total order broadcast is a broadcast where all correct processes in a system of multiple processes receive the same set of messages in the same sequence order. The total order is achieved using Mod-SMaRt [46] in BFT-SMaRt. The reason for total order is to support reliability in BFT system. We have the following reliability goals:

- **Agreement**: If any correct replica delivers a write or read operation m, then every correct replica delivers m.
- **Total Order**: If a correct replica has delivered write operations $m_1, m_2, \cdots, m_s$ and another correct replica has delivered $m'_1, m'_2, \cdots, m'_{s'}$, then $m_i = m'_i$ for $1 \leq i \leq min(s, s')$.
- **Liveness**: If a write operation m is submitted to $n-f$ correct replicas, then all correct replicas will eventually deliver m.
- **No Creation**: If a correct replica q delivers a message m with sender p, then m was previously sent to q by sender p.
- **No Duplication**: No message m is delivered by a correct replica more than once.

4) Fabric channels for access control: We leverage fabric Blockchain network for consent access control. A member only shares the digital consent with other members who are in the same channels.

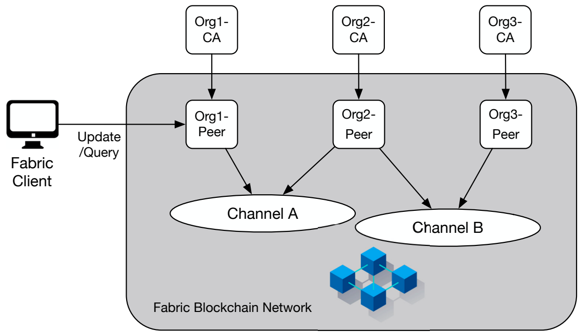

Figure 4. Fabric channels for access control.

Assuming we have 3 members in the fabric network, Org1, Org2, and Org3, as shown in Fig. 4. As Org1 and Org2 are in Channel A, and Org2 and Org3 are in the same Channel B, Org2 can access both data consent stored in the immutable ledgers.

C. ChiosLite *for Secure Data Collection*

For data collection, a lightweight ChiosLite client (lightweight version of ChiosEdge client) has been implemented which can be deployed on a low-cost operating system such as the Raspberry Pi. To authenticate the identity of an IoT device, ChiosEdge uses an ECDSA library for message authentication. All the IoT devices need to be registered and verified. And ChiosEdge servers gather only transactions sent by authorized devices. For sensitive data, ChiosLite additionally encrypted them on the IoT side.

In local organization, we tested on motion detection sensors that send the observation data m (message or ciphertext) to ChiosEdge client through ChiosLite, then ChiosEdge client distributes message m and hashcode $h(m)$ to all the peers. All the peer nodes run a BFT consensus algorithm to verify the message m. Finally, message m will be stored in all the peers' databases securely. Users who are interested in certain data can subscribe via an IoT device id. ChiosEdge servers push the data automatically to the subscribed users.

For other organizations, a user needs to get consent and permission from investigators. A detailed data consent smart contract and how investigators offer the permission are given in the following sub-sections, V-D and V-E.

D. Data Consent

Our data consent chaincode comprises the following eight primary operations for dealing with the fabric client and Blockchain ledger, as shown in Fig. 5. Consents are saved in the ledger and retrieved using the user's identity or the ledger's record ID. Each permission specifies a start and end date for lawful access. The data owner may rescind permission at any time, but it must be authorized by at least half of the investigators (data governance).

As every data owners have the right to ask questions or get all the sensitive information, the consent is an electronic document in a Blockchain for data owners themselves to fully control the sensitive records, surface the data ownership, and increase data confidence.

E. Decentralized Governance with (t, n) Adaptive Threshold Signature

We use the (t, n) adaptive threshold signature scheme as the core function for governing sensitive data as the access request needs to be signed before processing, the steps are shown in Fig. 6. The detailed steps are as follows:

1) Parameters generation phase: A group of investigators generates two key pairs, one for *yes* key and another is for *no* key. The *yes* key pair is (pri_i^{yes}, vk_i^{yes}) where a investigator keeps the pri_i^{yes} secret and publish its public verification key vk_i^{yes}. Similarly, the group of investigators generates the *no* key pairs (pri_i^{no}, vk_i^{no}). Every investigator in this step generates two key pairs *yes/no*, the *yes* key pair is for confirming that the message is valid and investigators can uses the key pri_i^{yes} to sign the message.

Operation Interfaces

```
// put the initial records into
// the chaincode Query.
(1) InitLedger();
// the variables includes recordID, contractID
// description, starttime, endtime,
// additionalterms, etc.
(2) CreateRecord();
// query a record by recordID or contractID
(3) QueryRecord();
// revoke a consent by recordID
(4) RevokeConsent();
// query a record by patient ID
(5) QueryRecordByUserId();
// query a record by consenting party ID
(6) QueryRecordByConsentingPartyId();
// get query result by a string
(7) getQueryResultForQueryString();
// query all records
(8) QueryAllRecords();
```

Figure 5. Primary operations for dealing with the fabric client and Blockchain ledger.

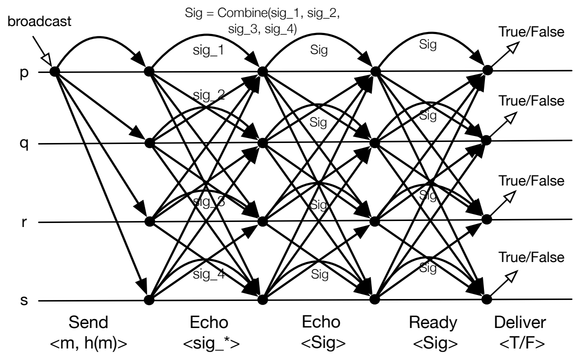

Figure 6. Reliable Broadcast and Threshold Signature for Decentralized Data Governance.

2) Transaction submitting phase: The investigator group consists of multiple n members $c1$, $c2$, ..., cn. We take four members as an example here. Fabric client calculates the hash code of the message $hash(m)$ or $h(m)$. Finally, the fabric client sends $<m, h(m)>$ to each of the investigator in the group.

3) Sign a signature: The group members received the message m and its hash value $h(m)$. First, they will check the content of the message m. If a member c_i confirms that this message is valid, then c_i uses its private *yes* key to sign the message and generates the share signature sig_i = $Sign(pri_i^{yes}$, $m)$. After that, c_i broadcasts the share signature sig_i to every investigators to generate the final signature Sig.

4) Make a consensus and deliver the result: Each of the investigator will receive the message m and $h(m)$, and all of them generate their signature sig_i after signing the signature. The node can first verify this share signature sig_i with the c_i's two public verification keys vk_i^{yes} and vk_i^{no}.

Then, every node (e.g., p, q, r, t) receives more than t thresholds (t, n) from the all investigators including himself (Phase 2, Echo(sig_*)), then they can run the combination algorithm to recover the final signature Sig = $Combine(sig_1$, sig_2,..., $sig_t)$. Finally, the each of them can verify the final signature with the public key pairs pk^{yes} or pk^{no}, and start to make consensus.

(i) Echo the <Sig> to all the investigators, if they received a Quorum of the same Sig, then goes to the Second phase.

(ii) In Byzantine reliable broadcast, the phase of Ready is for *totality*. The reason for this step is for amplifying the Ready message and deliver fast.

(iii) If the investigators received $f+1$ of Ready messages, then deliver the message m (True/False) to the Fabric client.

After this final signature is verified by the yes or no public key $true/false = Verify(m, fsig, pk^{yes})$ (pk^{no}), all the investigators can determine if this message *m* can be submitted to the Blockchain ledger or not. In addition, they will keep a copy of these results in the logs for further evidence.

VI. Implementation

ChiosEdge consists of a Java library and a Python library with about 25,000 lines of new code. We utilize the BFT-SMaRt consensus protocol written in Java as the underlying consensus engine. ChiosLite, written in Python, consists of an ECDSA cryptographic library for message authentication, which runs in cheap IoT devices such as Raspberry Pi. We deployed our data consent in AWS Managed Blockchain [1] (Hyperledger Fabric) with a customized smart contract written in Golang. We implemented a NodeJS API gateway module and a Terraform automation development tool with about 20,000 lines of code in total. We propose to use an adaptive threshold signature (ATS) which is written in Python for decentralized data governance. We also use gRPC [2] for underlying data transmission between different languages.

VII. Evaluations

A. Experimental Settings

For the local evaluation, we utilize an Intel(R) Xeon(R) Gold 5117 CPU with 28 cores and Ubuntu 16.04.6 LTS with the kernel version Linux 4.4.0-142-generic. To evaluate ChiosEdge in a wide area network (WAN) context, we referenced some results from our previous work and deployed it on Amazon EC2 with up to 31 consensus nodes and 25 client nodes (running up to 1,200 clients in total). By default, each node is a compute-optimized c5.2xlarge type with 8 virtual CPUs (vCPUs) and 16GB of memory. Additionally, we also test the performance on a variety of hardware configurations using the general-purpose t2.medium type with two vCPUs and 4GB memory. We evaluate our protocols in both LAN and WAN settings, where the LAN nodes are uniformly distributed on four Dell Servers in the school data center and the WAN nodes are randomly distributed across different AWS regions.

B. Latency in ChiosEdge *Module*

We evaluate the latency of Reja in both the WAN and LAN environments and reference some results from our previous Chios [26] pub/sub system. ChiosEdge is installed locally on four Dell servers to operate secure broadcast and BFT consensus. The network latency is quite low in LAN environments compared to WAN environments; the main overhead is caused by the BFT consensus mechanism, client-side encryption techniques, and database operations (e.g., read/write operations). In the WAN settings, as we deploy 4 to 31 nodes in the AWS cloud with different locations, the network latency is increased due to the transmission overhead.

1) Latency of Write: We set Blocksize to one in both the WAN and LAN settings and operate four BFT nodes to tolerate one Byzantine node. We repeated the test $3,000$ times with a 1KB write request. As illustrated in Fig. 7, the delay of a write request ranges between $33ms$ and $90ms$. The peak latency frequency is between $30ms$ to $40ms$.

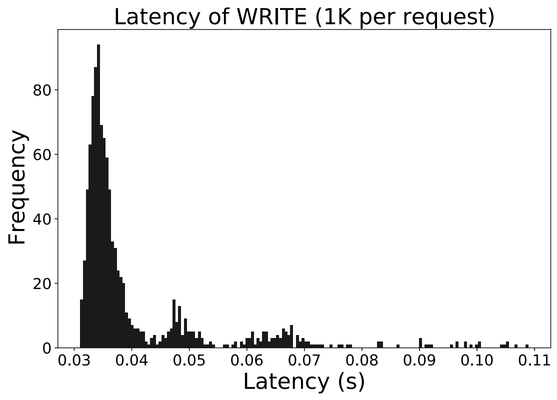

Figure 7. Latency of write per 1K request in LAN ($f = 1$).

We evaluate the latency of write with BlockSize=1 in WAN settings when $f = 1, 5,$ and 10, which means n equals to $4, 16,$ and 31 respectively, the result is shown in Fig. 8. The result shows that the latency in WAN is 8 times than it in LAN. It takes 0.4s average time for one write request. Here, f refers to faulty nodes and n refers to the total nodes.

2) Latency for Read: In the LAN settings, we read a 1K transaction stored in the Leveldb 1000 times. As illustrated in Fig. 9, most of the latency of a read request ranges between 30ms and 65ms, and more than 70% of latency are between 30ms to 40ms. A small part of latency ranges between 90ms to 100ms is caused by BFT consensus overhead and system internal scheduling.

3) Latency for multi-Write: In real settings, the system inevitably deploys thousands of IoT devices. We evaluate 50 to $50,000$ multi-write operations simultaneously in LAN settings. The result, as shown in Fig. 10, grows linear intuitively and theoretically, but with more write requests, the less the latency is. For example, when we run $50,000$ multi-write operations, it takes 107.47 seconds, we found that the average processing time for one request is about 0.002s. The reason is the throughput of ChiosEdge supports up to 30,000 transactions per second in LAN, as shown in Fig. 11.

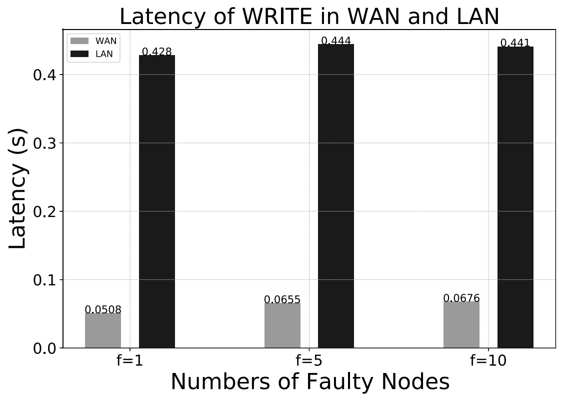

Figure 8. Latency of write in WAN and LAN. f refers to faulty node(s)

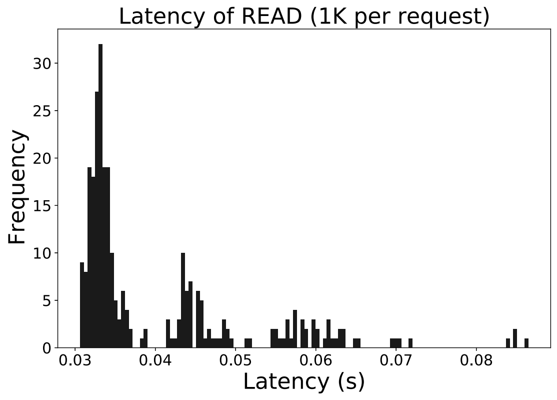

Figure 9. Latency of read per 1K request in LAN ($f = 1$).

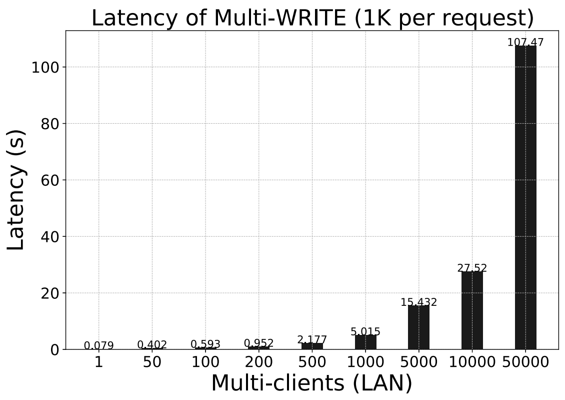

Figure 10. Latency of multi-write per 1K request.

C. *Throughput of* ChiosEdge *Module*

We also evaluate the throughput of ChiosEdge with BlockSize=1 in the LAN and WAN settings when $f = 1, 5$, and 10, which means n equals to $4, 16$, and 31 respectively.

Throughput in the LAN and WAN with 500 clients. We evaluate throughput using up to 31 servers and 500 clients both in LAN and WAN settings. As illustrated in Fig. 11, we find that the throughput for both WAN and LAN degrade when the number of servers increases. But in WAN settings, the throughput still can reach up to 18.7 kops/s (18,700/s). The maximum throughput in LAN reaches up to 37.31 kops/s.

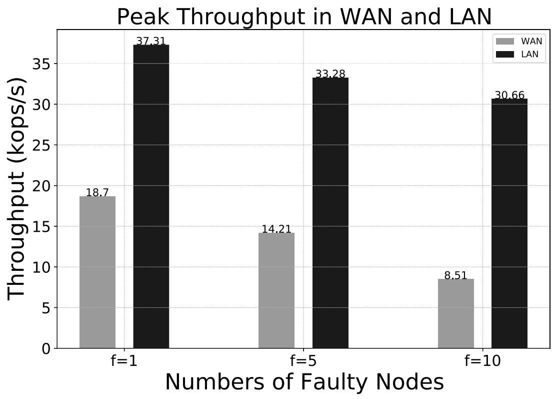

Figure 11. Peak Throughput in WAN and LAN. f refers to faulty node(s)

D. *Endpoints Response Time*

We propose to use permissioned Blockchain for recording data digital consent to surface data ownership. We deployed our consent smart contract in the AWS Managed Blockchain network. After deploying the API and Blockchain correctly, all endpoints (URL link) can be accessed via GET/POST requests. It includes user registration, consent registration, query by consent user ID, and so on. We conduct 10 times for one endpoint and take the average values as the results in Table. I (T_w refers to response time in WAN). In addition, some requests' response times, such as *query* (GET), are affected by the number of data consents stored in the Blockchain ledger. Query all consent endpoint takes about 6 seconds which should be noted that we have more than 30 consents stored in the immutable ledger. Apparently, the time required grows as the number of stored consent increases.

Table I
ENDPOINTS RESPONSE TIME.

Request	**Method**	$\mathbf{T}_w$
User Register	POST	70ms
Consent Grant	POST	133ms
Query Consent by UserID	GET	136ms
Consent Revoke	GET	120ms
Consent (query all consent)	GET	6.64s

VIII. DISCUSSION

We evaluated the latency and throughput of write and read in the WAN and LAN settings. We also evaluated the response time of all the endpoints in the AWS Managed Blockchain for the data consent module. We found that the overall latency is relatively higher than in non-BFT systems. But as we mentioned, Reja framework provides a modular framework allowing trade-offs between functionality, security, and efficiency. As the data is stored in decentralized edge nodes, the core component ChiosEdge brings real trust to the data. A BFT consensus protocol can tolerate Byzantine faults among the edge devices. A malicious user can hardly compromise half of the nodes at the same time. We suggest deploying your edge nodes in the WAN settings and setting the number of nodes to 4 or 7, even though the latency is 8 times higher than it in the LAN.

IX. CONCLUSIONS

The centralized data hub is vulnerable to being attacked and compromised, we propose Reja, a novel, secure, and decentralized IoT framework with permissioned Blockchain and an intrusion-tolerant BFT system in which the core components are reliable broadcast and BFT consensus. Reja can run over multiple edge devices to distribute copies of IoT data in replicas to avoid single-point-of-failure and Byzantine faults.

ACKNOWLEDGEMENT

We gratefully acknowledge the support of the NSF through grant IIP- 1919159. We also acknowledge the support of Chao Liu and Andrew Weiss.

REFERENCES

[1] https://aws.amazon.com/managed-blockchain/.
[2] https://grpc.io/.
[3] https://hyperledger-fabric.readthedocs.io/en/release-2.2/access_control.html.
[4] https://www.axios.com/facebook-data-533-million-leak-bda53583-363a-4e4a-bc38-b147c3e12a8c.html.
[5] https://www.forbes.com/sites/ajdellinger/2021/04/03/personal-date-of-533-million-facebook-users-leaks-online/?sh=3ee35bde717c.
[6] https://www.kaspersky.com/about/press-releases/2019_iot-under-fire-kaspersky-detects-more-than-100-million-attacks-on-smart-devices-in-h1-2019.
[7] https://www.zdnet.com/article/iot-vendor-wyze-confirms-server-leak/.
[8] H. Abdullah, W. Garcia, C. Peeters, P. Traynor, K. R. Butler, and J. Wilson. Practical hidden voice attacks against speech and speaker recognition systems. *arXiv preprint arXiv:1904.05734*, 2019.
[9] Z. Ba, S. Piao, X. Fu, D. Koutsonikolas, A. Mohaisen, and K. Ren. Abc: Enabling smartphone authentication with built-in camera. In *25th Annual Network and Distributed System Security Symposium, NDSS 2018*, 2018.
[10] A. Bessani, J. Sousa, and E. E. Alchieri. State machine replication for the masses with bft-smart. In *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, pages 355–362. IEEE, 2014.
[11] P. Bhanupriya, S. Gauni, K. Kalimuthu, and C. Manimegalai. A modified hybrid blockchain framework for secured data transaction. In *Journal of Physics: Conference Series*, volume 1964, page 042040. IOP Publishing, 2021.
[12] S. Birnbach, S. Eberz, and I. Martinovic. Peeves: Physical event verification in smart homes. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 1455–1467, 2019.

[13] M. Brandenburger, C. Cachin, R. Kapitza, and A. Sorniotti. Blockchain and trusted computing: Problems, pitfalls, and a solution for hyperledger fabric. *arXiv preprint arXiv:1805.08541*, 2018.

[14] C. Cachin et al. Architecture of the hyperledger blockchain fabric. In *Workshop on distributed cryptocurrencies and consensus ledgers*, volume 310, pages 1–4. Chicago, IL, 2016.

[15] Y. Cao, Y. Sun, and J. Min. Hybrid blockchain–based privacy-preserving electronic medical records sharing scheme across medical information control system. *Measurement and Control*, 53(7-8):1286–1299, 2020.

[16] M. Castro, B. Liskov, et al. Practical byzantine fault tolerance. In *OsDI*, volume 99, pages 173–186, 1999.

[17] Z. B. Celik, G. Tan, and P. D. McDaniel. Iotguard: Dynamic enforcement of security and safety policy in commodity iot. In *NDSS*, 2019.

[18] G. Crow, R. Wiles, S. Heath, and V. Charles. Research ethics and data quality: The implications of informed consent. *International Journal of Social Research Methodology*, 9(2):83–95, 2006.

[19] L. Cui, S. Yang, Z. Chen, Y. Pan, Z. Ming, and M. Xu. A decentralized and trusted edge computing platform for internet of things. *IEEE Internet of Things Journal*, 7(5):3910–3922, 2019.

[20] Z. Cui, X. Fei, S. Zhang, X. Cai, Y. Cao, W. Zhang, and J. Chen. A hybrid blockchain-based identity authentication scheme for multi-wsn. *IEEE Transactions on Services Computing*, 13(2):241–251, 2020.

[21] M. Daghmehchi Firoozjaei, A. Ghorbani, H. Kim, and J. Song. Hy-bridge: A hybrid blockchain for privacy-preserving and trustful energy transactions in internet-of-things platforms. *Sensors*, 20(3):928, 2020.

[22] H. Desai, M. Kantarcioglu, and L. Kagal. A hybrid blockchain architecture for privacy-enabled and accountable auctions. In *2019 IEEE International Conference on Blockchain (Blockchain)*, pages 34–43. IEEE, 2019.

[23] H. B. Desai, M. S. Ozdayi, and M. Kantarcioglu. Blockfla: Accountable federated learning via hybrid blockchain architecture. In *Proceedings of the eleventh ACM conference on data and application security and privacy*, pages 101–112, 2021.

[24] A. Dorri, S. S. Kanhere, R. Jurdak, and P. Gauravaram. Blockchain for iot security and privacy: The case study of a smart home. In *2017 IEEE international conference on pervasive computing and communications workshops (PerCom workshops)*, pages 618–623. IEEE, 2017.

[25] A. Dorri, S. S. Kanhere, R. Jurdak, and P. Gauravaram. Lsb: A lightweight scalable blockchain for iot security and anonymity. *Journal of Parallel and Distributed Computing*, 134:180–197, 2019.

[26] S. Duan, C. Liu, X. Wang, Y. Wu, S. Xu, Y. Yesha, and H. Zhang. Intrusion-tolerant and confidentiality-preserving publish/subscribe messaging. In *2020 International Symposium on Reliable Distributed Systems (SRDS)*, pages 319–328. IEEE, 2020.

[27] S. Fan, H. Zhang, Y. Zeng, and W. Cai. Hybrid blockchain-based resource trading system for federated learning in edge computing. *IEEE Internet of Things Journal*, 8(4):2252–2264, 2020.

[28] D. Ferraiolo, D. R. Kuhn, and R. Chandramouli. *Role-based access control*. Artech house, 2003.

[29] A. Gervais, G. O. Karame, K. Wüst, V. Glykantzis, H. Ritzdorf, and S. Capkun. On the security and performance of proof of work blockchains. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 3–16, 2016.

[30] A.-G. Gheorghe, C.-C. Crecana, C. Negru, F. Pop, and C. Dobre. Decentralized storage system for edge computing. In *2019 18th International Symposium on Parallel and Distributed Computing (ISPDC)*, pages 41–49. IEEE, 2019.

[31] W. He, M. Golla, R. Padhi, J. Ofek, M. Dürmuth, E. Fernandes, and B. Ur. Rethinking access control and authentication for the home internet of things (iot). In *27th {USENIX} Security Symposium ({USENIX} Security 18)*, pages 255–272, 2018.

[32] C. V. Helliar, L. Crawford, L. Rocca, C. Teodori, and M. Veneziani. Permissionless and permissioned blockchain diffusion. *International Journal of Information Management*, 54:102136, 2020.

[33] S. Herwig, K. Harvey, G. Hughey, R. Roberts, and D. Levin. Measurement and analysis of hajime, a peer-to-peer iot botnet. In *Network and Distributed Systems Security (NDSS) Symposium*, 2019.

[34] D. Johnson, A. Menezes, and S. Vanstone. The elliptic curve digital signature algorithm (ecdsa). *International journal of information security*, 1(1):36–63, 2001.

[35] A. A. E. Kalam, R. E. Baida, P. Balbiani, S. Benferhat, F. Cuppens, Y. Deswarte, A. Miege, C. Saurel, and G. Trouessin. Organization based access control. In *Proceedings POLICY 2003. IEEE 4th International Workshop on Policies for Distributed Systems and Networks*, pages 120–131. IEEE, 2003.

[36] M. A. Khan and K. Salah. Iot security: Review, blockchain solutions, and open challenges. *Future generation computer systems*, 82:395–411, 2018.

[37] D. Kumar, K. Shen, B. Case, D. Garg, G. Alperovich, D. Kuznetsov, R. Gupta, and Z. Durumeric. All things considered: an analysis of iot devices on home networks. In *28th {USENIX} Security Symposium ({USENIX} Security 19)*, pages 1169–1185, 2019.

[38] S. Kumar, Y. Hu, M. P. Andersen, R. A. Popa, and D. E. Culler. {JEDI}: Many-to-many end-to-end encryption and key delegation for iot. In *28th {USENIX} Security Symposium ({USENIX} Security 19)*, pages 1519–1536, 2019.

[39] S. Osborn. Mandatory access control and role-based access control revisited. In *Proceedings of the second ACM workshop on Role-based access control*, pages 31–40, 1997.

[40] J. Park and R. Sandhu. The uconabc usage control model. *ACM transactions on information and system security (TISSEC)*, 7(1):128–174, 2004.

[41] A. Reyna, C. Martín, J. Chen, E. Soler, and M. Díaz. On blockchain and its integration with iot. challenges and opportunities. *Future generation computer systems*, 88:173–190, 2018.

[42] K. M. Sadique, R. Rahmani, and P. Johannesson. Towards security on internet of things: applications and challenges in technology. *Procedia Computer Science*, 141:199–206, 2018.

[43] G. Sagirlar, B. Carminati, E. Ferrari, J. D. Sheehan, and E. Ragnoli. Hybrid-iot: Hybrid blockchain architecture for internet of things-pow sub-blockchains. In *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pages 1007–1016. IEEE, 2018.

[44] R. Sandhu and Q. Munawer. How to do discretionary access control using roles. In *Proceedings of the third ACM workshop on Role-based access control*, pages 47–54, 1998.

[45] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu. Edge computing: Vision and challenges. *IEEE internet of things journal*, 3(5):637–646, 2016.

[46] J. Sousa and A. Bessani. From byzantine consensus to bft state machine replication: A latency-optimal transformation. In *2012 Ninth European Dependable Computing Conference*, pages 37–48. IEEE, 2012.

[47] H. Yu, J. Lim, K. Kim, and S.-B. Lee. Pinto: enabling video privacy for commodity iot cameras. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 1089–1101, 2018.

[48] E. Yuan and J. Tong. Attributed based access control (abac) for web services. In *IEEE International Conference on Web Services (ICWS'05)*. IEEE, 2005.

[49] E. Zeng and F. Roesner. Understanding and improving security and privacy in multi-user smart homes: a design exploration and in-home user study. In *28th {USENIX} Security Symposium ({USENIX} Security 19)*, pages 159–176, 2019.

[50] W. Zhou, Y. Jia, Y. Yao, L. Zhu, L. Guan, Y. Mao, P. Liu, and Y. Zhang. Discovering and understanding the security hazards in the interactions between iot devices, mobile apps, and clouds on smart home platforms. In *28th {USENIX} Security Symposium ({USENIX} Security 19)*, pages 1133–1150, 2019.

[51] S. Zhu, Z. Cai, H. Hu, Y. Li, and W. Li. zkcrowd: a hybrid blockchain-based crowdsourcing platform. *IEEE Transactions on Industrial Informatics*, 16(6):4196–4205, 2019.

[52] S. Zhu, H. Hu, Y. Li, and W. Li. Hybrid blockchain design for privacy preserving crowdsourcing platform. In *2019 IEEE International Conference on Blockchain (Blockchain)*, pages 26–33. IEEE, 2019.

[53] S. Zou, J. Xi, G. Xu, M. Zhang, and Y. Lu. Crowdhb: A decentralized location privacy-preserving crowdsensing system based on a hybrid blockchain network. *IEEE Internet of Things Journal*, 2021.