Smart-Contract Enabled Decentralized Identity Management Framework for Industry 4.0

Akanksha Dixit
City, University of London
London, United Kingdom
Akanksha.Dixit@city.ac.uk

Asif Waqar
University of West London
London, United Kingdom
waqar.asif@uwl.ac.uk

Muttukrishnan Rajarajan
City, University of London
London, United Kingdom
R.Muttukrishnan@city.ac.uk

***Abstract*—The emergence of smart sensors and autonomous system technology in industrial sector is getting tremendous research attention. Industry 4.0 is heavily using connected machine paradigm to automate, track and maintain processes. However, a major challenge in Industrial Internet-of-Things(IIoT) remains managing the burgeoning number of sensors. The centralized architecture for managing these devices is raising concerns. In this work, we propose a novel proof-of-concept framework for managing identity and access management policies for IIoT. The new architecture is a smart contract enabled and blockchain based decentralized life-cycle and access management system. It is backed by decentralized storage technology InterPlanatery FileSystem (IPFS). Our architecture is decentralized and scalable unlike the state-of-art IIoT management architectures designed for clouds which are centralized.**

***Index Terms*—Industry 4.0, Internet of Things, Blockchain, Smart contracts, Identity-management.**

I. Introduction

With increased automation in industrial processes, machine-to-machine communication is becoming pervasive. Since, IoT provides a connected paradigm of intelligent devices that can sense the manufacturing processes closely, it is being widely exploited in industries like manufacturing, supply chain management and predictive maintenance [21]. This wide use of such IoT devices brings with itself a large number of management challenges. Whilst there are several traditional IT enterprise identity management solution available in the market, they are not suited for an Industrial Internet-of-Things(IIoT) setup. Most of the solutions are heavily centralized around cloud architecture which creates a bottleneck in real-time data access and enabling access control policies for smart devices [6]. When identities are stored in a centralized location they become a potential attack vector, the confidential information is prone to data breach and manipulation. A single breach not only has digital fatality but can have serious repercussions in a physical environment [2].

Industrial facilities are dynamic environments as compared to home or office workspaces. They have several mobile entities and access control of these devices is shared among various management parties unlike home environment which has a single controller. Sensors installed in such industrial facilities gather highly sensitive data and this initiates the need for secure management practises of such devices. In a conventional IIoT environment, the security of such sensitive data is at high risk due to the possibility of a single point of failure. To overcome this problem there is a need to develop a framework based around a decentralized architecture [14].

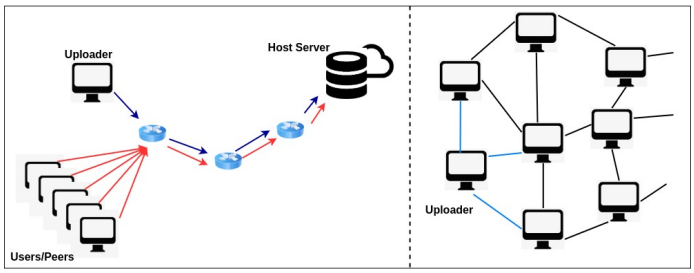


Figure 1: Client-Server Architecture Vs Decentralized Architecture

The blockchain technology has gained immense popularity in recent times due to their decentralized nature. It is known to be a decentralized, peer-to-peer technology which consists of cryptographically linked blocks of data storing transactions in the network [16]. These transactions represent the exchange of information among peers and in order to ensure authenticity, each transaction is added to the blockchain after a rigorous consensus mechanism thus making it a suitable decentralized solution [7] [8]. Decentralized systems promote trust and integrity in a system where many key-coordinators are managing a set of resources. Figure 1 illustrates the difference in data/command propagation both the models. In a client-server model, the flow of information is always towards or away from a central server which stores all the information the system. Whereas, in a decentralized system, each device in the network is a node hosting and consuming data simultaneously. Even if a number of nodes fail, the network keeps functioning unlike a centralized system where server failure means the network coming to a halt.

Although blockchain as a distributed ledger is capable of storing a certain number of transactions but its efficiency decreases as the amount of data stored on ledger increases. Increased data storage on blockchain increases the synchronization time among the peers and negatively affects scalability. Currently, the chain is growing at a rate of 1MB per block every 10 min in Bitcoin [20]. Therefore, it is not viable technically to store a large amount of data on blockchain

ledger. This becomes an even bigger issue when dealing with IIoT devices which tend to produce large volumes of data in a very short amount of time. The amount of data increases exponentially and the sensitivity of the data dictates that this information needs to be disseminated at the earliest. To overcome this issue, our proposed architecture is built using blockchain in conjunction with Inter-Planetary File system (IPFS). IPFS is a content addressed peer-to-peer distributed file system [3]. A file stored in IPFS has a unique hash that is derived from it's content. This makes the file secure from tampering by an unknown entity. IPFS eliminates the need of centralized storage entity in our architecture.

A. Case Study: Supply chain and warehouse maintenance

To understand the functionality and assess the performance of the proposed architecture for identity and access management, we are taking a use case from manufacturing sector. There are several reasons for using the IPFS backed blockchain solution for designing identity and access management in IIoT scenario. Firstly, there is a need for a decentralized identity management architecture given the large number of sensors that need to be managed. A large number of devices will try to connect simultaneously to the server for access control permissions, sending multiple request at a single point of time to central server. This server can be physically hosted on a different continent, affecting the response rate, hence resulting in potential degradation of performance. Secondly, security risk and trust management issues are reduced by storing identity related information and access control policies in a decentralized framework instead of using a centralized architecture or distributed one where the owner still remains a single entity.

Boeing, a leading aviation equipment manufacturing company is using IoT technology in factory floors to optimize production operations by improving predictive maintenance of the manufacturing sites [5]. The efficiency of such industrial organizations depends on managing downtime and shipment delays. And IoT plays a key role to monitor the processes closely. However, the company today relies on third-party cloud services to store operational and sensor data [9]. While using cloud infrastructure may seem an efficient tool in these scenarios, the problems associated cannot be ignored. The facility operational data and sensor data is confidential and must be protected against attacks to maintain accountability of critical components in aviation industry.

B. Contributions

In this work we propose a novel architecture based on decentralized technologies namely, blockchain and IPFS. The proposed architecture uses blockchain technology to manage identities and access control to these smart devices in the network. The main contributions of this work are:

- We implement a novel proof-of-concept decentralized identity management framework, which is based on a smart contract enabled blockchain architecture that is supported by distributed storage, IPFS.
- We show how our framework's performance being at-par with current state-of-art architecture and yet provide identity management and security in an industrial setting. It completely forgoes centralization and associated risks.
- We show that our decentralized framework can achieve a stable throughput and can be easily scaled up in enterprise setting which demands security and scalability.

The rest of this paper is organized as follows: Section II describes the state-of-art and related work in identity management for industrial IoT. Section III gives the architectural overview and component details with interaction workflow. Section IV discusses the experimental settings and presents the results. Section V provides a security assessment of our system and Section VI concludes the paper with ideas on future work.

II. Preliminaries

A. State-of-the-Art

Researchers across the world are exploring new technologies to make the Internet-of-Things scalable, secure and autonomous. The goal is to develop techniques so that the devices can communicate securely with minimum human intervention. A secure identity management framework is fundamental to achieve device-to-device communication for authentication and access management purpose with least human intervention. The present state-of-the-art primarily uses cloud based identity management infrastructures [10]. The major processing tasks and storage is delegated to cloud network. S. Hurrow et al. [12] describes an IdM technique for cloud based IoT. In this work, the *identity manager* is a component of cloud infrastructure that creates and stores digital identities associated with each device in cloud database, retrieving it during the device authentication process. Identity creation for new devices and removing existing devices is also managed via *identity manager*.

In the present IdM frameworks, there are two main methods of creating identities for IoT devices, first by giving a username-password for devices and second by using PKI (Public Key Infrastructure) to create digital certificates corresponding to each device. In both the methods device identities are stored in a central server and the task of authentication and access control is entrusted on a centralized entity. Often the authentication information like username and password are stored in plain-text in databases. This lead to a recent large-scale attack launched by Mirai botnet which infected up to 600k devices to successfully take down many online services [1]. The reason for attack was simple, using default username-passwords, attackers formed a large army of IoT devices to launch one of the largest DDoS (Distributed Denial of Service) attacks. Hence, public-key cryptography is useful to create identities and binding these identities to the physical attributes of devices.

B. Related Work

Some work related to distributed identity and access management of IoT devices are proposed by researchers. Novo [17] describes a generalized architecture for access management

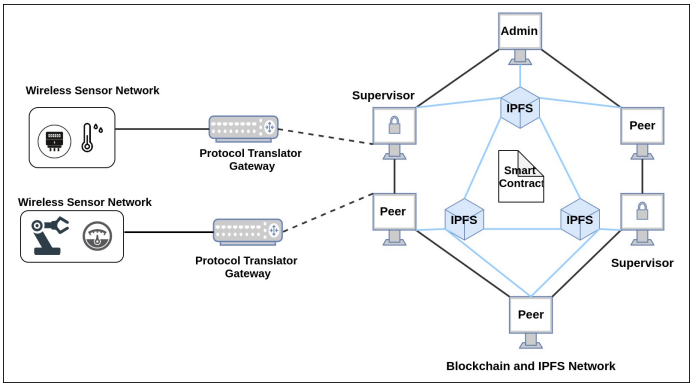


Figure 2: Decentralized Architecture for Industrial IoT. Each industrial facility hosts blockchain ledger and IPFS file system as a local copy.

in IoT using smart contract as enabler for defining access policies. This work mainly describes managing access policies for IoT devices by an intermediary interface. However, no identity creation and management is described in this work. Also, resource access policies are stored on the blockchain platform that can impact the performance of the architecture. Omar et al. [18] focuses on identity management in a general scenario specifically ownership management and transfer of IoT devices. They present a framework for identity creation and ownership transfer for IoT devices using blockchain.

This work relies on a global registry for storing identities created for IoT devices. Hammi et al. [11] proposes a scheme for a decentralized authentication in IoT network. In this work, secure virtual zones containing a few IoT devices are proposed and the devices in these zones trust each other. The inter-zone communication requires authentication which is achieved using smart contracts. However, these trust-bubbles creates a siloed architecture for sensor communication, in which authentication requests are directed to one manager node in the bubble.

Using smart contracts written in a turing complete language, Papadodimas et al. [19] presents implementation of DApp for providing data monetization. The devices can buy and sell data by leveraging the smart contract functionality. Li et al. [13] introduces the idea of using multiple satellite chains that form interconnected but independent sub-chains of a single blockchain network. These chains interact with each other with the help of specific 'validator' node. Each satellite chain implements a business logic. Mell et al. [15] proposes a federated authentication of user to a relying party without a third party authenticated service. The authentication is achieved between the relying party and user through a smart contract logic.

The above discussed works presents a number of approaches taken to decentralize the IoT infrastructure using the upcoming blockchain technology. However, most of them rely on blockchain ledger for data storage which can impede the scalability of the system with increased volume of IoT devices in the network. Device authentication procedure remains inexplicit in most of the cases, which as we discussed is an important feature to achieve device-to-device communication.

III. Proposed Architecture Overview

In the proposed work, blockchain acts as a decentralized identity management authority that is maintaining the mapping of identities to public keys. Smart contract provides the necessary logic to help in provisioning and revocation of keys of the IoT devices. The blockchain used in our architecture is a consortium blockchain setup by a group of participating parties to form a network of trust. The participating parties agree on a set of logical business operations, recorded in the smart contract. These set of operations govern the autonomous functioning of the network. Each party acts as a blockchain node and maintains a copy of ledger. The majority of policies and relevant metadata is stored on the IPFS while the blockchain only stores corresponding file hash as immutable transactions in the ledger. Every identity creation, deletion and policy update is logged on blockchain as a transaction. Access control policies defined in the network are retrieved from the local copy of the ledger.

IoT devices are resource constrained and hence they cannot act as a blockchain node. Therefore, in our architecture, the *Wireless Sensor Network (WSN)* formed by sensors is not included in blockchain network and interact with the later through a *protocol translator gateway (PTG)*. *PTGs* act on behalf of IoT devices to communicate with blockchain network and translate protocol between *WSN* and *PTG*. *Supervisors* are nodes that provision IoT devices in the network and manage their identity and access control.

A. System architecture

Figure 2 shows the components of the proposed architecture. The architecture consists of the following entities:

1) *Administrator Node*: The administrator node is a special node that initiates the blockchain network and invites other nodes to be a part of the network. This node deploys the smart contract that defines the functionalities of the entire system. Since this node deploys the smart contract in the blockchain it receives the address of this contract on the network. Later, this node shares the address with supervisor nodes in the network which allows them to interact with the smart contract.
2) *Blockchain Network*: The blockchain network in the proposed architecture is a consortium blockchain as it mimics the real world industrial scenario. Several participating parties adopt this model to promote trust and traceability in the system. A consortium blockchain allows fair participation of engaging parties as compared to private blockchain. Furthermore, it also has better transaction throughput and low setup cost when compared to a public blockchain. As a result, it offers advantages of both the types of blockchain and still achieves decentralization. The consensus in this network is achieved using a multi-party voting system. Each node

Table I: Table of Symbols

Symbol	Explanation
K^S_{pub}	Public Key of the Supervisor
K^S_{pri}	Private Key of the Supervisor
K^D_{pub}	Public Key of the Device
K^D_{pri}	Private Key of the Device
τ_S	Supervisor ID
τ_D	Device ID
$Sig_{K^S_{pri}}(\tau_S)$	Digitally Signed Supervisor ID
$Sig_{K^D_{pri}}(\tau_D)$	Digitally Signed Device ID
$\Im$	Ticket with device metadata stored in IPFS

can store transaction history as ledger and access global IPFS storage to access the identity and access related policies for IoT devices.

3) *Smart Contract*: The smart contract contains the core logic of the architecture in the form of its functions. A set of operations that allows entities in the network to provision identities to IoT devices, managing their access control and later revoking identities are defined in the smart contract. These operations can be triggered by the smart contract transactions sent by participating nodes in the network. A security feature is introduced in the contract function that authorizes only the registered supervisor nodes enabling them to send transaction against those functions. This feature restricts unauthorized access request load on the smart contract.
4) *Supervisor Nodes*: Any trusted peer node in the network can register itself as a supervisor node. Once these nodes are registered as supervisor node with the smart contract they can now register IoT devices in the blockchain platform. Supervisor nodes have higher computational and storage power and hence they can choose to host a copy of the blockchain. However, if these nodes are some lightweight node in the system they can choose not to host the ledger. Even in that case, nodes registered as supervisors can bring IoT devices on blockchain system. Each IoT device should register itself with atleast one registered supervisor node. It can also be registered under multiple supervisor nodes. That will ensure robustness in case of any faulty supervisor node.
5) *Wireless Sensor Network (WSN)*: The IoT sensors installed in facility form a WSN that interacts with blockchain network through gateway in the architecture. These sensors have limitations in terms of memory, power and computational capabilities. Each IoT device in the network is uniquely identified by its *SensorID*.
6) *Protocol Translator Gateway*: Since IoT sensors are resource constrained devices they use low power communication protocols like MQTT [1](Message Queuing Telemetry Transport) and CoAP (Constrained Application Protocol) within *WSN*. But blockchain network uses RPC protocol among it's peers. In order to provide interoperability between the blockchain network and IoT sensor network *PTGs* are used. They convert incoming MQTT requests from sensors to JSON-RPC message format before sending it to a supervisor node and vice-versa.
7) *IPFS File System*: IPFS file system is a content-addressable peer-to-peer file system. On uploading a file to IPFS, a hash of the content of the file is returned. Using IPFS in our architecture provides two benefits, first it removes the need of a centralized storage and second the content of the file is cryptographically secured using hashing mechanism. Any tampering to a file stored on IPFS can be detected and traced back.

[1]http://mqtt.org/

B. Smart Contract Functions

The functions defined in the smart contract are as follows. The symbols used in explanation are defined in Table I.

- *RegisterSupervisor*(K^S_{pub}, τ_S): to register any node S with Supervisor_ID τ_S as supervisor.
- *RegisterDevice*(K^D_{pub}, τ_D, K^S_{pub}): to register a device D with help of a supervisor S .
- *AuthenticateSupervisor*(K^S_{pub}, $Sig_{K^S_{pri}}(\tau_S)$): before a supervisor S can register/deregister a device, it needs to authenticate itself.
- *AuthenticateDevice*(K^D_{pub}, $Sig_{K^D_{pri}}(\tau_D)$): before a device can request/access a resource or even give a command to other IoT device, it needs to authenticate itself.
- *AddResourceControl*($ResourceList$): to add access control for a resource list of a device.
- *PermitResourceAccess*($D1$, $D2$, R): permit a device $D1$ to access a resource R hosted by device $D2$.
- *RemoveSupervisor*(K^S_{pub}, τ_S): to deregister a node S as supervisor.
- *RemoveDevice*(K^S_{pub}, τ_S): to deregister a device D from the network.

C. System Interaction and WorkFlow

1) Network Setup: In this phase, the consortium blockchain network is setup between the organizations involved in an agreement to maintain the network. Trusted peers are added in this phase to help maintain the distributed ledger. These peer nodes also host file-system for IPFS since they have high computational and storage capabilities. The administrator node in the network deploys the smart contract on the blockchain. This smart contract defines all the logical functions to register entities and policies to grant access to the IoT devices in the network.

Once the smart contract is deployed, the administrator node receives the blockchain address of this smart contract which it stores in a *Setup* file in IPFS and hash of this file is stored on blockchain. The address of this smart contract can be queried by any valid node from the IPFS which will allow them to send transactions to smart contract.

2) Registration of the entities: Any trusted node in the network added by consortium can register itself as a supervisor. To register, the node generates a public-private key pair (K^S_{pri},K^S_{pub}) using ECDSA algorithm which identifies

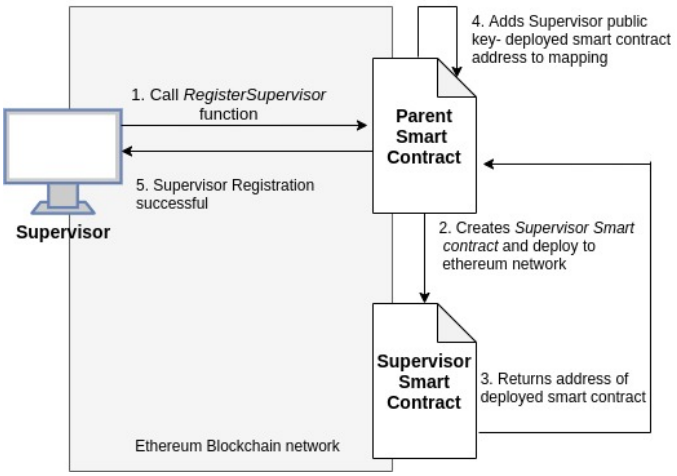


Figure 3: Registration of Supervisor

it uniquely in the network. The Keccak hash function [4] is used to generate a 20 byte supervisor ID τ_S from the public key. As shown in Figure 3, the supervisor sends a transaction to function *RegisterSupervisor* defined in the smart contract. This function deploys a smart contract of the registered supervisor and obtains address of deployed smart contract. The deployed supervisor's smart contract contains supervisor's metadata which includes supervisor's public key, ID and an array of devices registered under the supervisor. The parent smart contract will contain mapping of every registered supervisor's public key to the address of their respective deployed smart contract. This completes the registration of Supervisor node.

Figure 4 shows process of device registration. For device registration, the device will posses a public-private key pair (K^D_{pri},K^D_{pub}) and device ID τ_D derived from it. The device will send a registration request to the nearest supervisor, with its public key and τ_D. The supervisor will first invoke function *RegisterDevice* in the parent smart contract, this function calls another function *AuthenticateSupervisor* which takes as input K^S_{pub} and signature of supervisor over τ_S. From the received public key, supervisor's smart contract will be retrieved taking address from parent smart contract mapping. The received signature will be validated against the τ_S stored in supervisor's smart contract. Only a registered supervisor can add any IoT device in the network. Once validated, the supervisor will be allowed to create a *device ticket*, $\Im$ that will be stored in IPFS as shown in Figure 5. This $\Im$ contains all the metadata related to the device which includes K^D_{pub}, τ_D, hashed device attributes $H(A)$ and resource list associated with this device.

$$A = A_1||A_2||A_3.....A_n$$

$$H(A) = UniqueDeviceFingerprint$$

Device attributes are parameters like manufacturer details, firmware version, hardware version, location etc. All these attributes hashed together gives a unique fingerprint of the

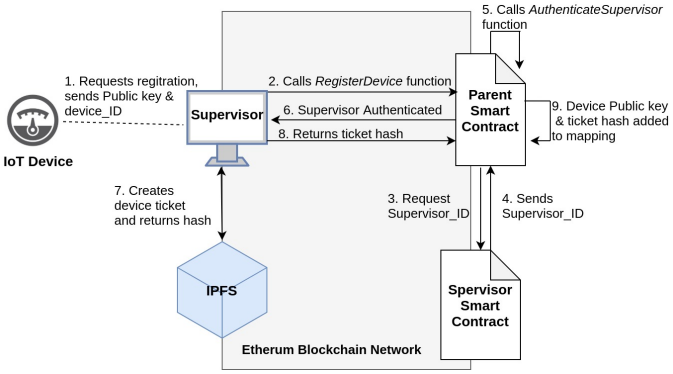


Figure 4: Registration of IoT Device

device which prevents any tampering of the device. On uploading the $\Im$ in IPFS, a hash string is returned. This ticket hash is stored in a mapping in parent smart contract along with K^D_{pub} and a dictionary of supervisors under which this device is registered as a device can be registered under more than one supervisor. This marks complete registration of a device in the system.

3) Add Resource policies: Supervisors can add resource access policies for IoT devices registered under them. After a device is registered under a supervisor, the access policies can be defined for that device. These policies can be defined in several ways. In our work, all the resources of a device are listed in their respective $\Im$. To set the access permissions, the supervisor first authenticates itself as explained in the device registration process. After successful authentication, supervisor is given access to $\Im$ location in IPFS to set the resource access policies of the device. The new hash of the ticket is updated in parent smart contract mapping which contains device public key mapped to $\Im$ hash in IPFS . This creates a distributed registry of device resources which can be accessed by any trusted node of the network to grant access to the devices.

Device Ticket IPFS Hash

```
{ deviceID : 0Xef289ma2h54t,
  resources: {
              "temp": D1, D4, D7;
              "humidity" : D2, D4,D9
              }
  device fingerprint: H(A) }
```

Figure 5: Format of device ticket $\Im$

4) Permit Resource Access: When a device $D1$ wants to access a resource R hosted by device $D2$, $D1$ sends a request containing it's signature over τ_D and public key to a PTG in the form of an MQTT message requesting the resource. The PTG transforms MQTT message to RPC-JSON request and forwards it to the nearest blockchain node. The node

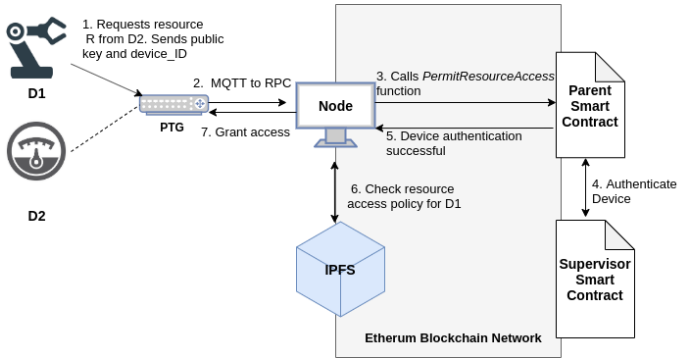


Figure 6: Permit Resource Access

calls *PermitResourceAccess* function defined in parent smart contract. This function first authenticates device $D1$ and if authenticated, allows the node to query the resource access policy for $D1$ stored in $\Im$ of device $D2$ in IPFS file system as shown in Figure 6. If $D1$ has access policy defined for resource R it sends the permission to PTG which sends it to $D1$. Since, the query is made to the copy of IPFS hosted on node, the process is completed immediately. In this case, there is no reliance to receive the information from a centralized storage.

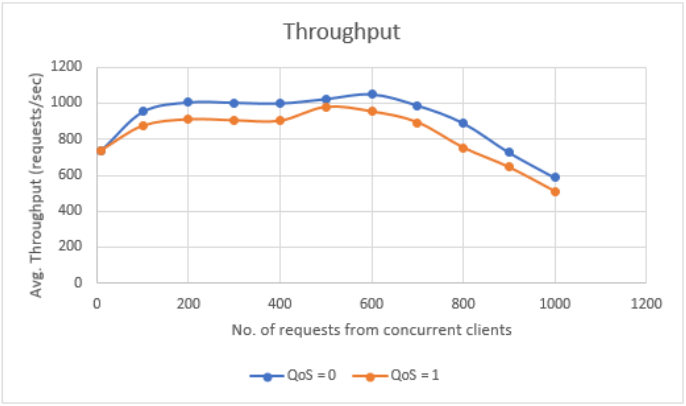
IV. Implementation and Results

A. Experimental setup

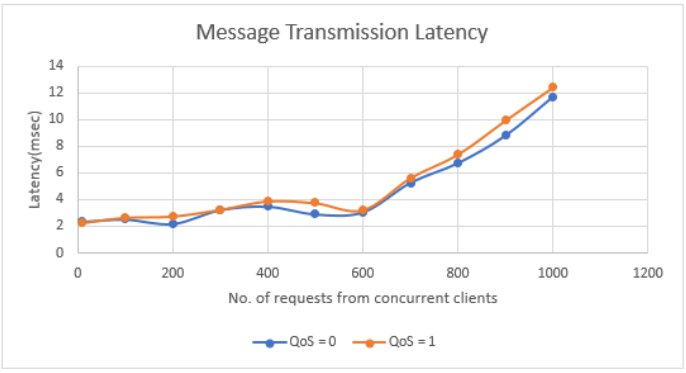
The experiment was setup on a machine installed with Linux operating system with the following hardware specifications: Intel® Core™ i5-7200U CPU@2.50GHz and 8 GiB RAM. A private Ethereum blockchain [2] was used to deploy smart contracts, perform transactions and run tests. An IPFS node [3] written in Go v1.12.0 was installed on the system. This node can be a part of IPFS cluster hosted in an industrial zone. In order to interact with the IoT devices and enable exchange of messages between the devices, a Mosquitto MQTT [4] message broker v1.6.9 from Eclipse foundation is installed on the testing machine.

In order to analyze the performance of the setup, an MQTT benchmark tool written in Python 3.6 called PyMQTTBench[5] from Python Packages is used. The tool was modified to incorporate authentication process via blockchain and IPFS. This benchmark tool helps to create several virtual IoT clients, simultaneously requesting resource access in the network. We chose to implement the IoT devices communication using MQTT as it is a lightweight protocol and closely resembles the communication requirement of an industrial setup where real time data is used to drive machines and take decisions. MQTT protocol works in a pub-sub model where the clients publish to a *topic*, another term for a specific data stream from sensors. Publishers are clients that publish the data to the broker in specific *topics* and subscribers are the clients that subscribe to the *topics* the application is interested in.

[2]https://www.trufflesuite.com/ganache
[3]https://ipfs.io/
[4]https://mosquitto.org/
[5]https://pypi.org/project/pymqtt/



(a) Throughput performed on subscribers requesting access control for a resource hosted on a device in the network through *PTG*



(b) Latency experienced in resource access message delivery for the tests performed in Figure 7(a)

Figure 7: Results

B. Performance Evaluation

In the experimental setup, we assume that IoT devices created as virtual clients in the benchmark tools try to subscribe to resources hosted by other devices in the network. In order to subscribe to these data streams, the client's request the nearest *PTG* which in turn forwards the request to the blockchain network. The interaction between blockchain network and IPFS node then follows as explained in Figure 6.

The goal of the benchmark is to evaluate the impact of number of concurrent client requests on the network in terms of first, delivered throughput i.e. message rate on the subscriber side and second, message transmission latency i.e. time required to transmit a message to the client. Both of these parameters are evaluated for levels QoS = 0 and QoS = 1. MQTT protocol defines QoS as Quality of service which is assigned for both publishers and subscribers. QoS = 0 level, also called "at most once" means that message may be delivered once or not at all and QoS = 1 level, also called "at least once" ensures that receiver gets the message at least once. This is achieved through an acknowledgement

exchange between subscriber and broker. In our experiment both publishers and subscribers are configured.

The test is executed as a sequence of three steps. In the first step, the network is setup, smart contract is deployed, IPFS node is initialized and keys are generated for supervisors and devices. In the second step, the supervisors are registered followed by registration of devices through a *PTG*. In the third step, MQTT broker is setup and virtual publishers start publishing data concurrently to a *topic* like *"/resource/temperature/D1"*. A large number of subscribers send resource access request simultaneously to the *PTG* for connecting to MQTT broker for *topic* subscription. Once the authentication is obtained from blockchain network with the help of device metadata on IPFS, the subscriber can connect to broker and subscribe to the *topic* to start receiving the requested data. Each test was run ten times and average values are taken.

Figure **??** shows the results for throughput for both QoS = 0 and QoS = 1. The throughput for client requests steadily increases from 750 requests served per second until it achieves a throughput of 1000 requests per second for approx 600 client requests in case of QoS = 0. Similarly, for QoS = 1, the throughput increases from 720 requests to 930 requests per second. However, the overall throughput for QoS = 1 remains smaller then QoS = 0 due to the fact that after each message delivery the broker waits for acknowledgement in QoS = 1 resulting in lower throughput values. The decrease in the performance is directly related to the increase in message transmission latency as shown in Figure **??**. The latency for QoS = 1 is always greater than that for case QoS = 0 due to the acknowledgment that takes place after each message exchange and authentication process completion in blockchain network. The latency in both the cases remain well below 4 msec until the number of requests are below 600 and further increases with increase in the number of requests.

A greater throughput and better latency can be achieved with higher number of *PTGs* in the network and faster authentication processing in blockchain and IPFS network. Nevertheless, the performance of the overall setup performs well with up to 1000 requests served per second for 600 clients. The performance can be considered acceptable considering that with the increase in *PTGs* the performance will improve. And a WSN will have several *PTGs* interacting with supervisors.

V. Security Analysis

Security is a prominent feature of any system architecture. And when it comes to a huge manufacturing plant, security cannot be negotiated as these are critical infrastructures. Our proposed architecture should be no exception to this. In our architecture there are four main entities that interact with each other to fulfil the objective. They are IoT devices, *PTGs*, Supervisor nodes and IPFS nodes. We identify the susceptibility of each of the components for following threats: Spoofing, Denial-of-service, Tampering, Man-in-the-middle attack and Repudiation.

Spoofing attack takes place when a malicious entity tries to impersonate another device in the network. The IoT devices in WSN and supervisor nodes are not susceptible to this attack as they possess a public-private key-pair, which they use in all the interaction with the blockchain. However, since IoT devices are not a part of blockchain network and their communication with Supervisor nodes depends on *PTGs*. A malicious *PTG* can change the information to and from IoT devices. To overcome this issue, the *PTGs* can be assigned signed certificates by a trusted authority in the network. Each IPFS node has it's unique identity created at the time of installation and hence can prevent spoofing.

Denial-of-service (DoS) attack is possible on IoT devices and to avoid it, the devices must be non-reachable from an open network. To ensure it, firewalls can be installed in the network to prevent any malicious entity from across the network to launch DoS attack. Also, the unique identity of devices can help mitigate DoS attack on the network. The blockchain network in general provides a certain level of security against malicious nodes. Moreover, since the blockchain in our architecture is a permissioned network hence, only trusted nodes are added in the network. This significantly improves the performance of the network as the resource intensive mining process is replaced by a voting mechanism.

Tampering of access control information can be done by a malicious entity posing as *PTG*, but as proposed earlier, a signed certificate can help mitigate the attack. Tampering of device metadata in IPFS and transaction record in blockchain in difficult because both are cryptographically protected. In IPFS, each change to the *device ticket* is stored on blockchain as a hash of the file and any tampering with the data will result in a different hash. Similarly, in blockchain each block is cryptographically linked to the previous block making tampering highly difficult.

Man-in-the-middle attack is launched on IoT devices to access the resources from the devices by sniffing packets communicated between client and server. However, since all the devices and supervisor nodes are pre-registered in the network, it will be difficult for an impersonator to launch this type of attack as any resource access request from a malicious device will fail the authentication process. Non-repudiation is a major asset of using blockchain technology as the sequence of events are logged in the form of transactions in the blocks. In our architecture, direct interaction between IoT devices is not encouraged, and this will prevent any spoofing attack by one device on another.

VI. Conclusion and Future Work

In this paper, we addressed the issues in current Industry 4.0 implementation where IoT technology is completely based on a centralized ecosystem hence causing scalability issues on a horizontal and vertical scale. The identity and access management software used in industry are designed around the legacy centralized architecture. This paper proposes a novel identity management solution which is completely decentralized and based on decentralized technologies like blockchain and IPFS.

Due to the resource constrained nature of the IoT devices, they are not made a part of blockchain network and communication with them is achieved via a *PTG*. This makes the integration of IoT technology and blockchain technology highly adaptable with no major changes to the existing infrastructure.

The goal of this paper was to provide a decentralized and secure architecture for managing identities of IoT devices in the network with no single point of failure. The architecture should also be scalable without overloading the existing network. Such an architecture has been proposed in this work with some preliminary results. The throughput scales up with higher number of supervisors in the network and does not require any modification in current network layout. In the future, we continue to work towards addressing aspects of authorization and authentication in a decentralized manner for the proposed scheme.

References

[1] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis, D. Kumar, C. Lever, Z. Ma, J. Mason, D. Menscher, C. Seaman, N. Sullivan, K. Thomas, and Y. Zhou. "Understanding the Mirai Botnet". In: *Proceedings of the 26th USENIX Conference on Security Symposium*. USENIX Association, 2017, pp. 1093–1110. ISBN: 9781931971409.

[2] B. Bencsáth, G. Pék, L. Buttyán, and M. Félegyházi. "Duqu: Analysis, Detection, and Lessons Learned". In: *ACM European Workshop on System Security (EuroSec)* (2012).

[3] J. Benet. "IPFS - Content Addressed, Versioned, P2P File System". In: *CoRR* abs/1407.3561 (2014).

[4] G Bertoni, J Daemen, M Peeters, and G. V. Assche. "The KECCAK SHA-3". In: *"Proceedings Eurocrypt 2013-32nd Annual International Conference Theory Applications Cryptograph*. Vol. 7881, pp. 313–314.

[5] M. Castillo. *Honeywell Is Now Tracking $1 Billion In Boeing Parts On A Blockchain*. https://www.forbes.com/sites/michaeldelcastillo/2020/03/07/honeywell-is-now-tracking-1-billion-in-boeing-parts-on-a-blockchain/. [Online; accessed 23-March-2020s].

[6] H. Choi, J. Song, and K. Yi. "Brightics-IoT: Towards Effective Industrial IoT Platforms for Connected Smart Factories". In: *2018 IEEE International Conference on Industrial Internet (ICII)*. 2018, pp. 146–152.

[7] K. Christidis and M. Devetsikiotis. "Blockchains and Smart Contracts for the Internet of Things". In: *IEEE Access* 4 (2016), pp. 2292–2303. ISSN: 2169-3536.

[8] T. M. Fernández-Caramés and P. Fraga-Lamas. "A Review on the Use of Blockchain for the Internet of Things". In: *IEEE Access* 6 (2018), pp. 32979–33001. ISSN: 2169-3536.

[9] C. Gutierrez. *Boeing Improves Operations with Blockchain and the Internet of Things*. https://www.altoros.com/blog/boeing-improves-operations-with-blockchain-and-the-internet-of-things/. [Online; accessed 23-March-2020s].

[10] U. Habiba, R Masood, M. A. Shibli, and M. A. Niazi. "Cloud identity management security issues & solutions: a taxonomy". In: *Complex Adaptive Systems Modeling* 2.1 (2014), p. 5. ISSN: 2194-3206.

[11] M.T. Hammi, B. Hammi, P. Bellot, and A. Serhrouchni. "Bubbles of Trust: A decentralized blockchain-based authentication system for IoT". In: *Computers Security* 78 (2018), pp. 126–142. ISSN: 0167-4048.

[12] S. Horrow and A. Sardana. "Identity Management Framework for Cloud Based Internet of Things". In: *Proceedings of the First International Conference on Security of Internet of Things*. 2012, pp. 200–203. ISBN: 9781450318228.

[13] W. Li, A. Sforzin, S. Fedorov, and G. O. Karame. "Towards Scalable and Private Industrial Blockchains". In: *Proceedings of the ACM Workshop on Blockchain, Cryptocurrencies and Contracts*. 2017, pp. 9–14. ISBN: 9781450349741.

[14] P. Mahalle, S. Babar, R.N. Prasad, and R. Prasad. "Identity Management Framework towards Internet of Things (IoT): Roadmap and Key Challenges". In: *Recent Trends in Network Security and Applications*. 2010, pp. 430–439. ISBN: 978-3-642-14478-3.

[15] P. M. Mell, J. F. Dray, and J. Shook. "Smart Contract Federated Identity Management without Third Party Authentication Services". In: *Open Identity Summit 2019*.

[16] S. Nakamoto. *Bitcoin: A peer-to-peer electronic cash system*. 2009. URL: http://www.bitcoin.org/bitcoin.pdf.

[17] O. Novo. "Blockchain Meets IoT: An Architecture for Scalable Access Management in IoT". In: *IEEE Internet of Things Journal* 5.2 (Apr. 2018), pp. 1184–1195. ISSN: 2372-2541.

[18] A. S. Omar and O. Basir. "Identity Management in IoT Networks Using Blockchain and Smart Contracts". In: *2018 IEEE International Conference on Internet of Things (iThings)*. July 2018, pp. 994–1000.

[19] G. Papadodimas, G. Palaiokrasas, A. Litke, and T. Varvarigou. "Implementation of smart contracts for blockchain based IoT applications". In: *2018 9th International Conference on the Network of the Future (NOF)*. Nov. 2018, pp. 60–67.

[20] A. Reyna, C. Martín, J. Chen, E. Soler, and M. Díaz. "On blockchain and its integration with IoT. Challenges and opportunities". In: *Future Generation Computer Systems* 88 (2018), pp. 173–190. ISSN: 0167-739X.

[21] L. D. Xu, W. He, and S. Li. "Internet of Things in Industries: A Survey". In: *IEEE Transactions on Industrial Informatics* 10.4 (2014), pp. 2233–2243.