IJIST
P ISSN : 2618-1630
E ISSN : 2909-6130

Accountable and Trustworthy IoT Networks, Based on Blockchain

Original Article

Abid Sultan[1], Yao Lin[1]
[1]School of Software, Dalian University of Technology, China.
*Correspondence: Abid Sultan and abidsultan006@gmail.com

Citation | Y. L. Abid Sultan, "Accountable and Trustworthy IoT Networks, Based on Blockchain," *Int. J. Innov. Sci. Technol.*, vol. 5, no. 1, pp. 94–110, 2023.
DOI | https://doi.org/10.33411/IJIST/2023050107
Received |Jan 29, 2023; **Revised** | Feb 25, 2023; **Accepted** | March 08, 2021; **Published** | March 09, 2023.

The term "Internet of Things" (IoT) refers to a situation in which intelligent things are linked to a network or the internet. IoT objects have become more prevalent over the past several years in many industries, and fields, and are now used in all facets of our life. The privacy of data is a crucial problem as the number of devices rises. Researchers in this discipline have employed a variety of strategies to address this issue. Regrettably, there is less accountability, data protection, and traceability with these solutions. In this study, a blockchain-based network architecture for accountability, privacy, and traceability is designed (TDA). Blockchain technologies are referred to as a distributed ledger of transaction records, which time-stamped information about a transaction's lifetime. Persistence, decentralization, and audibility are three of blockchain's key characteristics. The budget is reduced and efficiency is increased thanks to these characteristics. This study also discusses the performance of the suggested architecture in order to strengthen the TDA architecture.

Keywords: Blockchain, Accountability, Architecture and Traceability.

Author's Contribution
All authors have contributed equally.

Conflict of interest
The authors declare no conflict of interest in publishing this manuscript in IJIST.

Project details. NIL

Introduction

IoT is a recent trend, however, the concept of using computer systems and networks to observe or monitor objects has been for years. For instance, commercially successful programs to remotely monitor electricity grid meters have existed since the 1970s.Advances in wireless technology helped machine-to-machine (M2M) communication and offered solutions to commercial issues in 1990, which helped M2M gain popularity for allowing tools to be watched. M2M is reliant on specific networks and industry standards. The Internet Protocol (IP) standard is subsequently used to connect IoT components as a result. The first IP-supported device that can be turned on to start working and off to cease working over the Internet is the capability of the toaster.

The standards of our lives have changed as a result of modern technology. This is a result of technological advancements and innovations that certify items for interconnection over an IoT network. Making an informed decision is made easier with the help of the data gathered from this network which is Internet-of-things (IoT) [1].Cisco forecasts the IoTs as the assumption of people, places, and things, for making services available for usage by other objects. One illustration is the Thermostat-Nest installed in homes to adjust the temperature of the room remotely. These Internet of Things devices make life easier for homeowners.

IoT devices are often used in industries for maintenance. In order to acquire data that is useful for decision-making and to stop unneeded actions, sensors, and cameras are used for it. IoT is employed in daily activities, and at business, because it facilitates data collection and decision-making. According to the research by International Data Corporation (IDC), 20% of IoT companies leverage blockchain technology's foundational services, and 75% of all IoT manufacturers enhance the security features to make their products even smarter for customers [2].

The information records on instruments are exposed to attackers by abusing the components in connected IoT networks, this is due to the diversified services and IoT network integration. Moreover, this causes unauthorized access to the information. The communication protocols are used to distribute information among the IoT network's components. The "things," which range from wearables to massive machinery made of sensor chips, are different. [3] [4]. Nevertheless, the core Client-Server Model used to connect IoT components results in a variety of security concerns in IoT networks.

There have been numerous methods for traceability, data privacy, and accountability presented by researchers in recent years [4][5][6][7] but, the studies that have already been conducted still have significant holes in them. Blockchain technology, which has a number of capabilities to offer the answer to issues that arise in IoT network devices, has been developed in recent years. Blockchain keeps track of all IoT device lifespans. With Proof of Work, centralized third parties that control the entire network in a centralized design are eliminated. It is useful in addressing the IoT security issues shown in Table 1 [8] Consequently, the purpose of this article is to propose answers for the problems of accountability, data privacy, and traceability using the blockchain.

The IoT network is the subject of this research, which aims to provide user privacy and accountability for actions. Every time a user requests access to their data or potentially modifies a transaction record, their traceability must be guaranteed with respect to acceptable audibility. So, both the tools and their users are responsible for their actions.

1) How is the privacy of data safeguarded in IoT networks?
2) How are people in the IoT network given more control over their personal data?
3) How can the IoT network be made more secure and scalable?
4) How are requests and responses made through the IoT network tracked?
5) Create a decentralized Internet of Things network for TDA, scalability, and to prevent single-point failure.

This main goal of this paper is to develop a decentralized method for detecting traceability, data privacy, and accountability (TDA) within the IoT network architecture. Delivering user privacy and accountability for actions that take place inside the IoT network is the primary goal at hand. Every time a user requests access to their data or potentially modifies a transaction record, their traceability must be guaranteed with respect to acceptable audibility. So, both the tools and their users are responsible for their actions.

IoT devices play a significant role in the smart society, which includes hospitals and electricity generation facilities where data is shared for various tasks. In these categories, user information is more sensitive. Devices in resource-constrained environments have a limited amount of processing power, bandwidth, and memory. IoT devices connected to the internet and observed the surroundings to conduct high-level functionalities [9]. As shown, for instance, in Figure 1 Data is more vulnerable to attack by unauthorized access and may be misused by IoT network nodes because of the diverse connections between services and smart IoT components [10] [11] [12]. As a result, the need for privacy and data protection is growing in the IoT sector. Additionally, given the cloud-based environment of IoT, source tracing becomes a significant issue. For instance, a user might ask the source to act directly. Due to the direct connection, traceability is made simple.

Traceability is difficult while using the cloud, there is an indirect link between the devices. Due to the abundance of resources available for use in the cloud, this issue occurs. For instance, if a user requests data from a server remotely, the server will want to know where the request originated from as well as the various ways it was sent to the server. The end-user also requests to know the source of the data after obtaining the response.

It is significant to highlight that in such a cloud restrictions enjoinment, the data may be altered. IoT consequently increases traceability-related concerns. A case study of Abid and Amir, two brothers who reside in a smart home with their family in order to better grasp this issue, (equipped with IoT devices). Abid one day turned off his Smart AC before leaving the house for work. However, his brother Amir, who also departed for college, neglected to turn off the smart air conditioner after turning it on. When Abid returned from work at 6 o'clock, he saw that his smart AC with IoT capabilities was on. At whatever time he is at his office, Abid will be disappointed and demand to know which member of the home switched it on. If Amir linked to AC with a direct link as depicted in Figure 2, it would be simple to trace as indicated in Figure 3. There isn't a direct link between Amir and AC. Due to the abundance of resources in the cloud-constrained environment, it becomes difficult to trace. As a result, it is quite difficult to determine who is responsible for the cloud's services.

Protocol Use for Blockchain IoT Communication

In order to exchange data and transfer it to a storage medium for additional processing, the interconnection between IoT devices is necessary. But, these elements needed to speak a common language, known as the protocol, in order to communicate. Message Queue Telemetry (MQTT) and Constrained Application Protocol are the two most used IoT protocols, in contrast, even if there are many others (CoAP).

Message Queue Telemetry (MQTT)

This protocol, which was created by IBM, intends to connect small components to a network with constrained resources. Transmission Control Protocol is employed (TCP). Additionally, it employs three phases as companions for message dissemination. Deliveries of messages occur at most once. This indicates that the delivery of the message is not acknowledged. A message must be delivered at least once, as ensured by the phrase "at least once." On the other hand, the message can probably be repeatedly sandblasted. Exactly once: Message delivery and confirmation receipt between the sender and the receiver take a lengthy time. By doing this, the message is delivered only once.

Table.1 IoT issues solution with Blockchain features [8].

IoT Issues	Blockchain Characteristics							
	Decartelization	Persistency	Anonymity	Scalability or More Addressing Space	Resilient Backend	High efficiency	Transparency	Smart contract
Data Privacy	✓		✓					✓
Data Integrity	✓	✓						✓
Third-party	✓				✓	✓		
Trusted Data Origin	✓	✓					✓	
Access control						✓	✓	✓
Single Points of Failure	✓				✓	✓		
Scalability				✓				

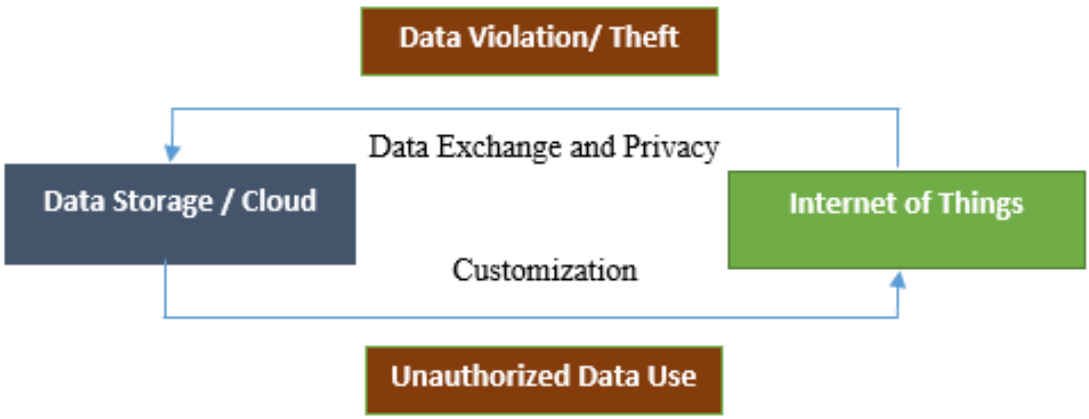

Figure 1. Violation of data in IoT environment [25].

Constrained Application Protocol (CoAP)

At the University of Bremen, CoAP was created in June 2014 to enable communication between Internet of Things (IoT) components with constrained processing and bandwidth. The Internet Engineering Task Force (IEFT) created CoAP, a Representational State Transfer protocol (RESTful) that utilizes HTTP and UDP (User Data Protocol) (Hypertext Transfer Protocol). Moreover, CoAP supports a wide variety of packet sizes, from 4 to 1024 bytes [13] [14]. CoAP contains the following four message classifications: Non-Verifiable-Message: This type does not receive an acknowledgment. Verifiable message: For this kind, an acknowledgment is sent. A user sent a connection request message repeatedly until the same message ID was returned as an acknowledgment. Reset message: This message is used to confirm a message that has not been verified or has been received. A message used to confirm the receipt of a confirmation message is known as an acknowledgment message.

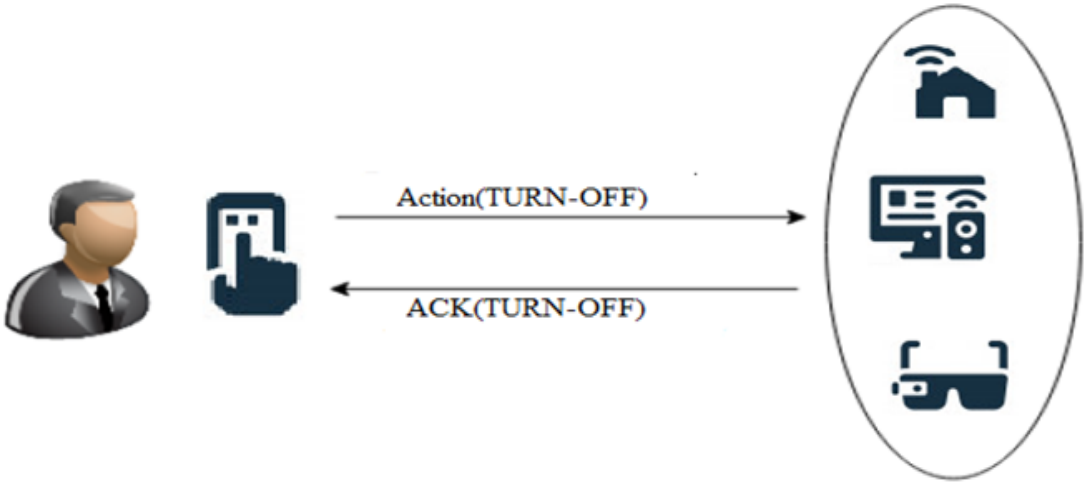

Figure 2. The direct connection between User and IoT Devices.

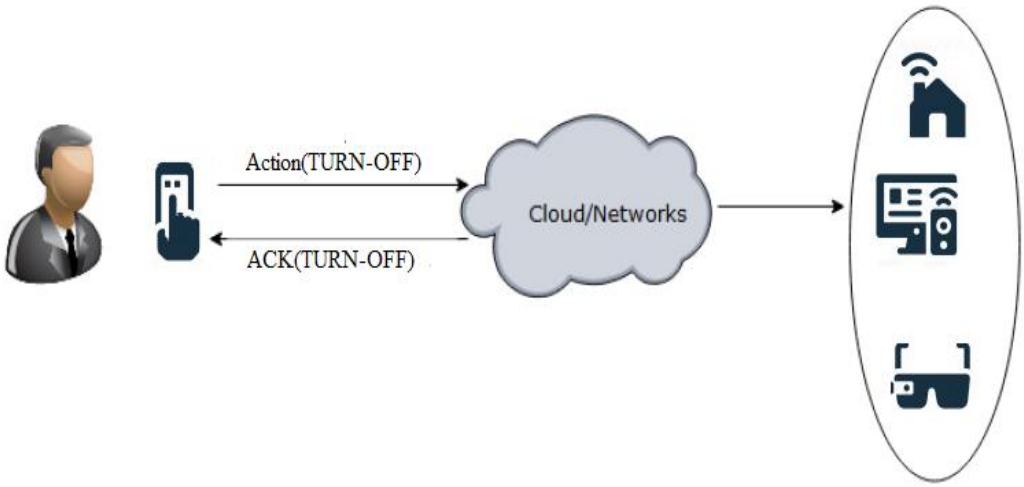

Figure 3. Indirect Connection through the cloud between User and IoT devices

REST

Representational State Transfer (REST) was introduced by Roy in 2000 [15]. REST is a substantial structure that emphasizes the critical standards that underlie contemporary Web program architectures. The web allows for a loosely coupled application architecture where the design of the architecture is primarily dependent on the available resources. PUT, POST, GET, and DELETE are some of the techniques used to access resources [15].

Related Works

Several academics have recently performed a study on the traceability, data privacy, and accountability of the intelligent IoT network. To become familiar with the shifting behavior of the IoT network environment, recognize well-organized public key sharing supervision as well as consistency assessment, avoid the network's participant nodes, believe the phony keys that are made publicly available, and ensure both the privacy and the effectiveness of data communication.

For various reasons, various researchers [1][9][3] have developed solutions for IoT data privacy. For instance, a Dynamic Trust Relationships Aware Data Privacy Protection (DTRPP) tool is described in crowd-sensing in mobile authors [16]. Each node must direct the dynamic evaluations to the public keys they give as well as the facing nodes. Several privacy standards are used to characterize and protect the data. The author in [17] devised a way to identify and stop assaults in communication networks, but it was unable to find any new attacks (e.g. zero-day exploits attacks). There is no surefire way to eliminate all security risks from a network, and

doing so is impracticable. To learn more about network nodes, attack targets are given priority based on where they are in the attack graph. In [18], Abazari and Madani proposed a threat-calculating weighted attack graph-based methodology. This is specifically intended to lessen risks and expenses. a multifunctional dynamic proactive threat reaction model.

A wireless sensor network Intrusion Detection System (IDS) utilizing a pattern comparison technique was suggested by Kalnoor and J. Agarkhed [19]. Pattern comparison explains unfavorable events based on pattern comparison defined by a set of signatures when a pattern matches an event, and a specified action is carried out determined by a set of signatures or rules. The IDS then analyses the quiet data and contrasts it with a large signature set. When there is a persistent discrepancy between the current and historical patterns, an alarm is generated.

For the ecosystem of smart vehicles, Ali Dorri et al. [20] developed a decentralized, protected BC-based architecture. Original Equipment Manufacturers (OEMs, or automobile makers), for example, collaborate to create an overlay network so they can connect with one another. Because nodes in the overlay are grouped together and only the cluster heads (CHs) are in charge of managing the BC and carrying out its primary function, these nodes are referred to as overlay block managers (OBM). A central broker is not required because transactions are broadcast to and confirmed by the OBM. Each vehicle has in-vehicle storage to keep privacy-sensitive data in order to safeguard user privacy (e.g., location traces). The owner of the car determines whether data should be shared with other parties in exchange for beneficial services and which data should solely be kept in the in-vehicle storage (and the granularity).

So, the originator has a greater right to manage the data during the communication process while working with the overlay vehicles. When a vehicle is physically separated from its associated OBM, latency increases.

Mandriva Banerjee and Kim-Kwang Raymond Choo [21] presented a method in which a central hub, where the datasets are kept and dispersed, maintains references of member repositories. Blockchain saved membership details like address, owner, and sharing guidelines. In other words, both the hub and all members have access to and maintain membership information. In order to maintain the Reference Integrity Measure (RIM) of datasets—which guarantees the integrity of the datasets—there is another chain of blocks. When datasets are publicly available, privacy is a key concern. The author emphasizes the need for an automatic tool that generates anonymous datasets prior to the announcement of these datasets in order to avoid the negative effects of any data privacy legislation and protect privacy. We also need to investigate the data sets' lifespan. It's possible that the owner of the underlying datasets won't always wish to release the dataset.

But, once a transaction is recorded on a blockchain, it cannot be changed or deleted. While being a robust security feature, this might not be useful for distribution if a record needs to be deleted. In the suggested dataset framework, the blockchain is responsible for maintaining the exclusive RIM. Hence, even if the RIM stays in the blockchain, datasets won't be accessible for a lengthy period of sharing.

All relevant studies show that nothing has been done to provide responsibility, traceability, and data privacy in the IoT network. Based on these premises, the goal of this article is to build a TDA-based blockchain architecture that offers data privacy and traceability in the IoT network. In Figure 4, the suggested architecture is displayed.

Proposed Architecture

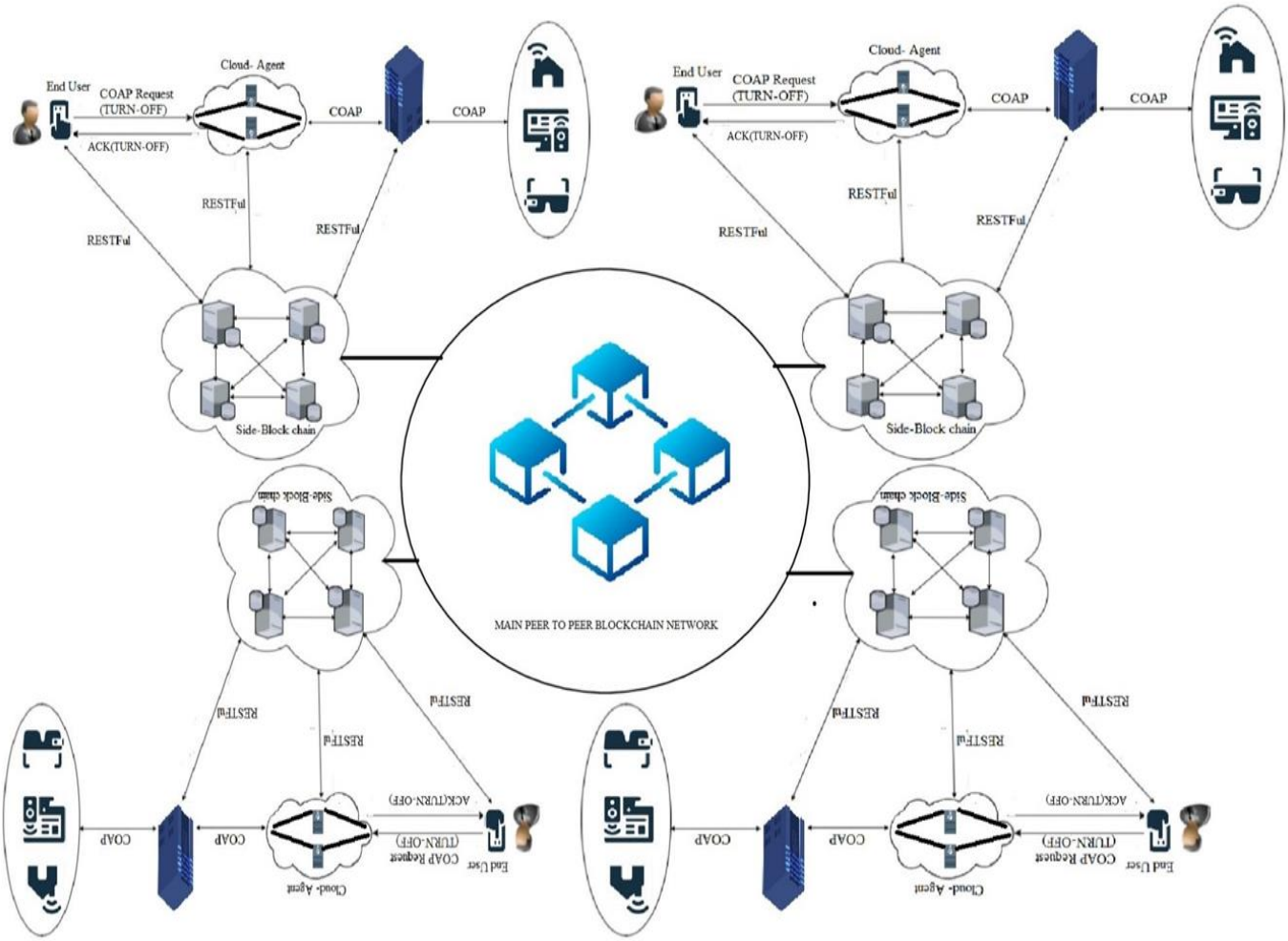

Figure 4. Proposed TDA Architecture

Data Privacy in the Proposed System

An IoT network architecture known as the "TDA" architecture is created. Figure 5 explains how the main blockchain is dissolving into side blockchains to ensure the privacy of data (SBC). One peer-to-peer IoT network is made up of these SBC IoT networks that are connected to one another. The main advantage of employing SBCs is that nodes in one SBC do not have the authority to participate in the transaction validation process in another SBC. Users control these SBCs based on their requirements. Every SBC has a unique set of network use cases and has the option of being added to or removed from the main blockchain. The end devices must first be identified by and integrated into the SBC IoT network in order to access information from that network. Access control and granting authorized access are made easier by SBC. Because all IoT records are visible to users, the privacy of the data may be jeopardized if only one blockchain is used. Nevertheless, the suggested architectural design will only allow a network entity to receive IoT data following a successful access request.

As an illustration, a network of SBC cameras gathers video data from the surroundings and adds it to a new block of this SBC. Lastly, its hash is saved in the primary blockchain's smart contract. Similar to the SBC network a sensor is attached to the new block and hash saved in the smart contract, another SBC network of sensors also monitor the surroundings. The validator node of this network must grant authorization for the SBC sensor network to

access data from the SBC cameras network. The validator node can finally see the network content of the camera after receiving authorization from it.

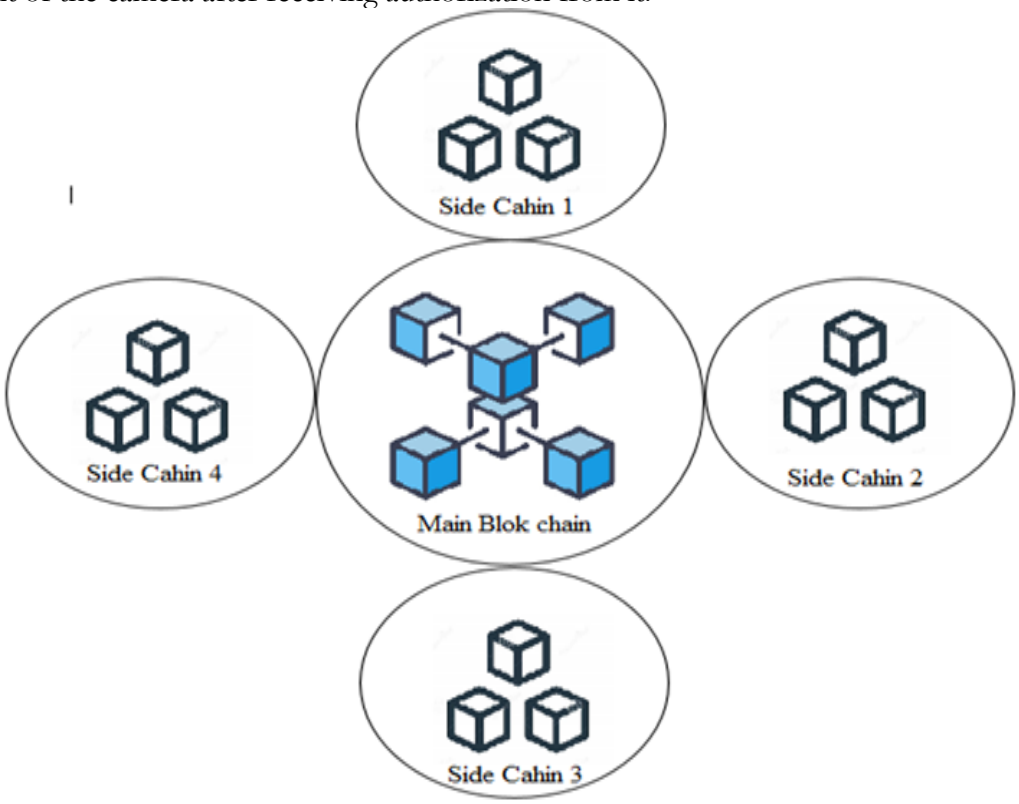

Figure 5. Blockchain IoT Data Privacy [8].

In Figure 6, a request initiator essentially has to join the main blockchain network by submitting an application as a member peer in order to access data or information from SBC. Together with sharing IoT data, the owner of the SBC adds the request initiator's public key and the amount of time needed for access permissions to be granted to that person to the SBC smart contract.

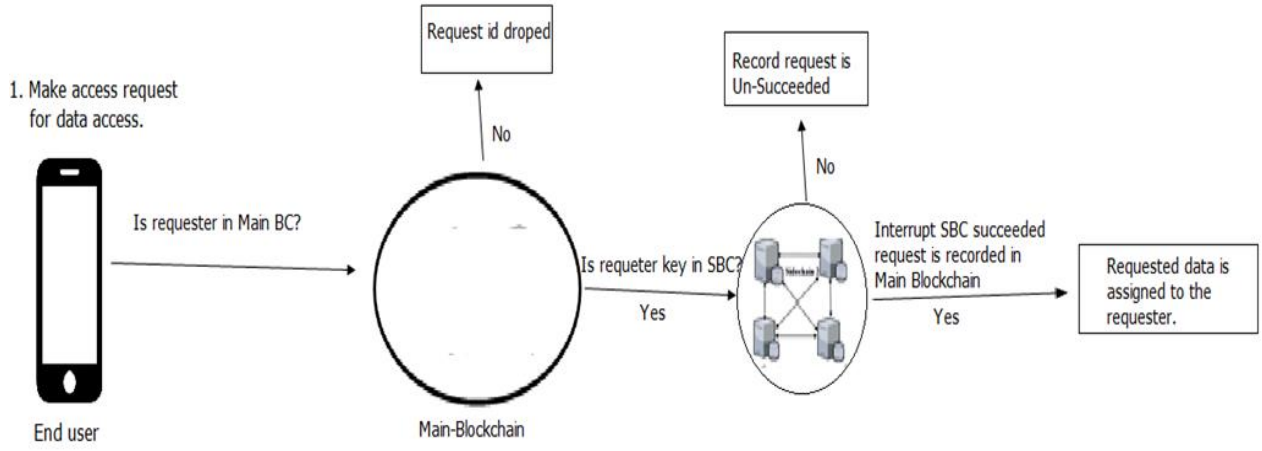

Figure 6. IoT Data Access Process

At this moment, the validation node also uploaded the public key to the blockchain in the list of genuine participants. The desired transaction is digitally signed by the initiator with the private key in order to gain access to IoT data. Figure 6 shows a process for accessing IoT data. By having the option to modify the access rules, the end user is able to decide with whom he wants to exchange IoT data. The smart contract deletes the appropriate key from the list of

legitimate requesters when a requester's request range exceeds the total number of unsuccessful requests.

Traceability in the System Under Discussion

The End-user (request initiator), a cloud-based agent, and the server are the major characters of the proposed TDA architecture, as shown in Figure 7 (that delivers services). The end user establishes a connection to the blockchain database and then logs information about its resources, including a time stamp, server or service provider URI, and a uniform resource identifier (URI). As the study focuses on recording the request/response lifecycle in the communication between IoT components in order to provide traceability.

The end user also sends a CoAP request and includes its URI in the desired payload. If the request is successfully made, the cloud-Agent will handle it and record the information in a distributed database. In order to request a blockchain, the cloud-Agent first establishes a new chain and logs the request's (To, From, Timestamp, Resource, and Method) database of blockchain information. The cloud agent utilizes a randomly generated message ID to verify the message's uniqueness, which must be done by comparing it to the message that the end user actually sent. The message received from the end-user was then modified by the clod-Agent to include its URI information before being transmitted to the services provider.

It is significant to note that there is a chance that a request message provided by the end user will go through various tracks and represent the number of cloud agents in the environment with partial resources. When a request message is received, the server establishes a connection and retrieves the message's (To, From, Time-stamp, Resource, and Method) data from the blockchain's database.

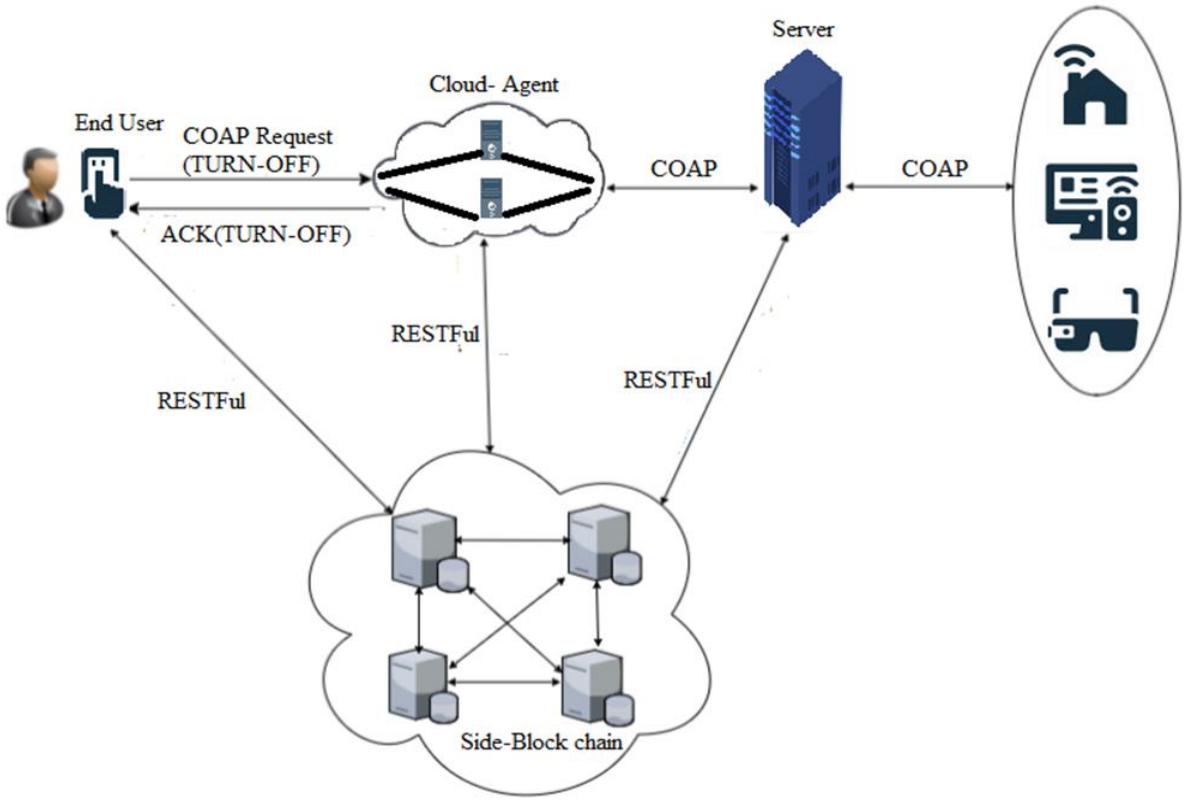

Figure 7. Traceability in Proposed TDA Architecture

Also, the server creates a fresh message and guarantees the originality of the data it receives from the cloud-Agent. To ensure uniqueness, the server employs a randomly generated message ID that is sent by the cloud-Agent. The request message must have the same message ID as the source in order for this to be completed as it travels through various IoT components.

A different chain known as a response chain is also created by the server and recorded in the blockchain. The server or service provider often uses the record as the source of the information, which is (To, From, Timestamp, resource, and method). Also, the server adds its URI to the reply message before sending the reply message status and the response contents to the cloud agent. As previously stated, it is possible in the cloud for the server's response message to take several routes and not be sent in the same manner that it arrived.

Any cloud agent in the cloud may therefore be able to intercept the response message before sending the request message. After receiving the response message, the cloud agent creates a message for a response with the information (To, From, Timestamp, resource, and method) inside the response chain and then records its source into the blockchain. And then use a message ID that was generated at random to match the original message ID. Also, the cloud agent included its URI in the server-replied message. Together with the status, the cloud agent also gave the end-user the response information. The cloud agent sends a message to the end user with the status of the response. The end user generates response data following receipt within the blockchain's response chain, which contains the original data (To, From, Timestamp, resource, and method). Additionally, a randomly generated message ID is compared with the answer message to ensure that the message ID is the same as the original.

By adding its URI to the message that originates from the cloud agent, the end-user also indicates how it acquired the message. The blockchain stores and records every piece of source data. The source blockchain's immutability makes it a legitimate technology that works best for source tracking or traceability. This is because the database is an append-only one, making it difficult to change source information once it has been stored because all focused activities from start to finish are recorded. The modules become transparent and accountable as a result of source tracking, as seen in Figure 8.

Responsibility for Requests Made

Each network user has access to the permanent record of all access requests, which ensures accountability for all requests. Consider the case where a member sells his data to a company. For a period of five days, the member is allowed access to the company. The organization will not be granted access if the owner terminates the rights of access before the expiration date. The company may provide a time-stamped record of any unsuccessful access attempts.

Smart Contracts' ability to program the blockchain facilitates transaction execution by ensuring that the contract code's requirements are met. Smart contracts are deployed in the blockchain using a specific address, and a transaction is signed digitally by nodes and addressed to the smart contracts in order to carry out the activity specified or programmed in a smart contract. However, the proposed TDA architecture includes conditions enforced by smart contracts for the transaction and access control policies.

Implementation and Results

The suggested architecture is implemented using Golang and Multichain. It enables software engineers and developers to create efficient and user-friendly software. The Golang third-party library, which provides support for both CoAP and server-side implementation, and is created by Dustin Sallings, is used to implement the suggested architecture.

The following four sections make up the implementation code.

Customer Side

According to Figure 8, end-user coding is done in Golang and enables the end user to connect to the distributed database of multichain, the decentralized blockchain. A function called Write Record stores the End User's Message-ID, To, From, Process, URI, and Time-stamp after interconnection. The request that contained the message data used to describe the End User URI (uniform resource identifier) was then passed to Cloud-Agent with the aforementioned data attached.

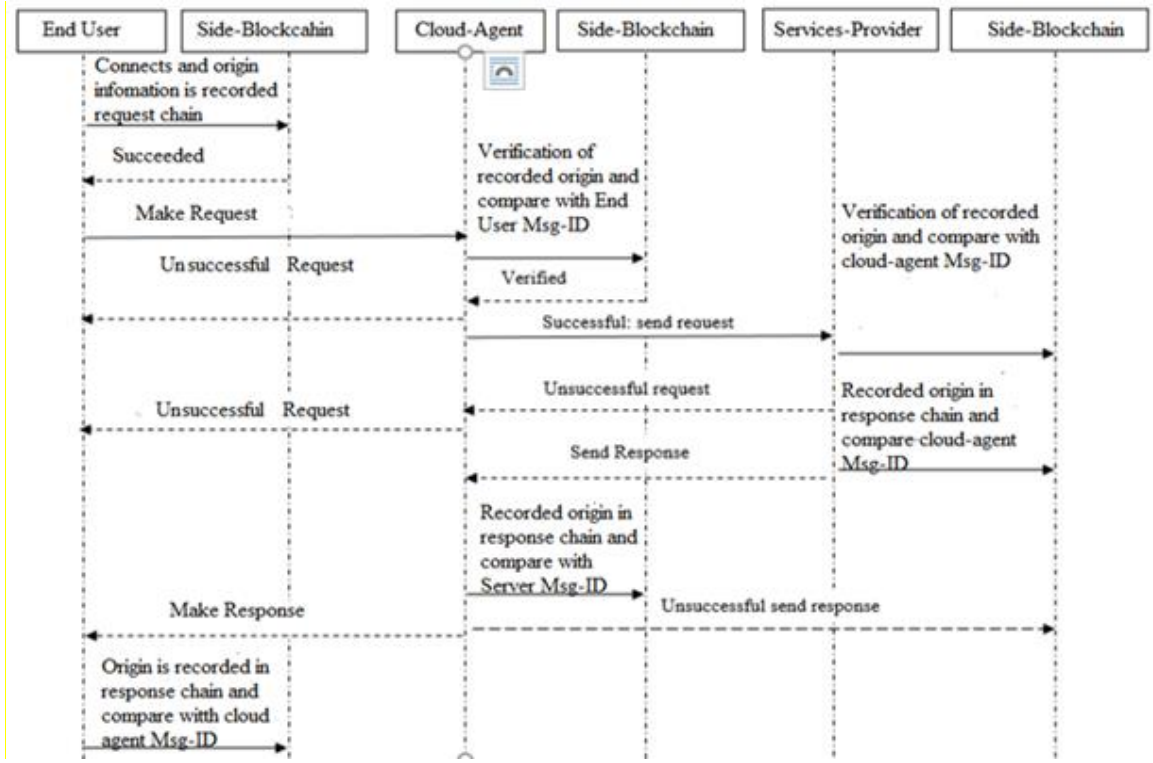

Figure 8. Sequence diagram complete communication among the roles as adapted from [22]

```
func main() {

  //CLIENT CONNECT AND MAKE REQUEST
  c, err := core.Connect(core.AgentAddr)
  if err != nil {
    log.Fatalf("Error: %v", err)
  }

  // create a record
  var actor core.MessageRecord
  actor.MessageID = core.CreateMessageID()
  actor.From = core.OriName
  actor.To = core.AgentName
  actor.Method = string(core.Method)
  actor.URI = core.GlobalURI
  actor.Timestamp = core.GetCurrentTime()
  err = core.WriteRecord(core.TableName_Request, actor)
  if err != nil {
    log.Printf("Error: %v\n", err)
    return
  }
```

Figure 9. End-user side code.

The Cloud-Agent Side

As seen in Figure 10, the cloud-agent side code is also written in the Golang language. The cloud-agent side of the code connects the cloud-agent to the multichain and uses the Write Record method to write source data to the multichain. When a user sends a request message, the clod-agent receives it and acknowledges it. Deliver it to the server at the end.

```
coap.FuncHandler(func(l *net.UDPConn, a *net.UDPAddr, m *coap.Message) *coap.Message {

    log.Printf("Received Message ID (%d)\n\n", m.MessageID)

    if m.IsConfirmable() {

      c, err := core.Connect(core.SvcproviderAddr)
      if err != nil {
        log.Fatalf("Error: %v\n", err)
      }

      var actor core.MessageRecord
      actor.MessageID = m.MessageID
      actor.From = core.AgentName
      actor.To = core.SvcName
      actor.Method = string(m.Code)
      actor.URI = m.PathString()
      actor.Timestamp = core.GetCurrentTime()
      err = core.WriteRecord(core.TableName_Request, actor)
      if err != nil {
        log.Printf("Error: %v\n", err)
      }

      req := core.BuildConReq(actor.MessageID, actor.URI, m.Payload)
```

Figure 10. Cloud-Agent side code.

Web Server Programming

Golang is also used to code the server for providing services, as shown in Figure 11. Once the request and server interlinks have been heard and recognized, the Write Record function is used to record the source information, such as the Message-ID, To, From, Process, URI, and Time-stamp. For message identification, the URI of the server or services provider is also supplied. The end user and cloud agent follow a similar process, and efficient code implementation is used to ensure traceability in the bidirectional communication between the end user and cloud agent as well as between the cloud agent and the services provider or server.

Setting up the Environment for Implementation

The hardware and software components that make up an implementation setup. Install Move Add Change (IMAC), a workstation, and a raspberry pie-2 are the hardware requirements for this practical implementation. IoT end-user component simulation is done using the IMAC. The workstation is for the cloud agent, and the raspberry pi server of type pie-2 is for server simulation. The following is a list of the components.

Local Environment Components

IMAC – End-User Device

1. Brand =IMAC
2. OS = Mac OS X 10.10Yosemite
3. CPU =Intel Core i5 CPU 3.0 GHz
4. Memory =16GB

Raspberry1pi-2: Cloud-Agent

1. Brand =Raspberry-Pi-2
2. Model=B
3. OS=Raspbian
4. CPU=800 MHz plus 3.0 GHz
5. Memory - 2GB

The workstation of Lenovo for the Server

1. Brand=M series2. OS=Windows 10
2. CPU=Intel Core i7 3.40 GHz
3. Memory=16GB

Workstation for Blockchain Nodes

1. OS=Ubuntu 16.04 Long-Term Support (LTS)
2. CPU=Intel Core i7 3.40GHz
3. Memory=8 GB RAM

Cloud

1. OS: Ubuntu 16.04 Long-Term Support (LTS) for every node
2. CPU=Intel Xeon 2.40GHz
3. Instance of type=t2.medium
4. Availability zone=us-east-b
5. Memory=8GB RAM

```
func main() {

  log.Printf("Service is running on %v\n", core.SvcproviderAddr)

  log.Fatal(coap.ListenAndServe("udp", core.SvcproviderAddr,
    coap.FuncHandler(func(l *net.UDPConn, a *net.UDPAddr, m *coap.Message) *coap.Message {

      log.Printf("Received Message ID (%d)\n\n", m.MessageID)

      var actor core.MessageRecord
      actor.MessageID = m.MessageID
      actor.From = core.SvcName
      actor.To = core.AgentName
      actor.Method = string(m.Code)
      actor.URI = m.PathString()
      actor.Timestamp = core.GetCurrentTime()
      err := core.WriteRecord(core.TableName_Response, actor)
      if err != nil {
        log.Printf("Error: %v\n", err)
      }

      if m.IsConfirmable() {

        req := core.BuildAckReq(*m)
```

Figure 11. Server-side code

Physical Composition

In both the local environment and the cloud, the structure is organized. The IoT components can communicate locally thanks to Ethernet technology. With the use of Ethernet technology, the server, raspberry pie-2, blockchain database (Multi-Chain), and end-user are all connected to a switch.

Prior to conducting any actual experiments, each code is assessed on the specific system. Each implementation code completes a particular function depending on the action it takes inside the system. Being a distributed database built using blockchain technology, three end nodes are required for analysis across the entire testing environment [15].

Data Gathering

On the end device, we use Apache J-Meter, an open-source performance measurement tool that supports many performance assessment metrics. Apache JMeter uses a library created by Dustin Sallings to support the CoAP protocol. A data set of various sizes, including 4, 8, 16, 32, 64, 128, 256, and 512 bytes, is also chosen for performance evaluation. 1, 20, 40, 80, and 160 users were chosen to ensure that the system can sustain a large number of users.

Measures of Performance

• Response-Time: The amount of time it takes for an End device to send a CoAP request—passed as final from the Agent to the Server—and to write the record to the Database. In milliseconds, it is respectable (ms).

• Throughput: The entire amount of data that is sent via the network per second is known as throughput. Transactions per second are used to measure it.

• Pay-load1size: The total amount of data sent via the network together with the CoAP request, expressed in bytes.

• Users Quantity: the total number of users who concurrently request a system. It is employed to evaluate scalability.

Performance Assessment

Figure 12 shows the payload sizes ranging from 4, 8, 16, 32, 64, 128, 256, and 512 are gathered together with the appropriate reaction times, as shown in Figure 11. The results showed that reaction times for pay-loads ranging from 4 to 128 bytes were between 380 and 384 ms. Nevertheless, 256 and 512 should have faster response times.

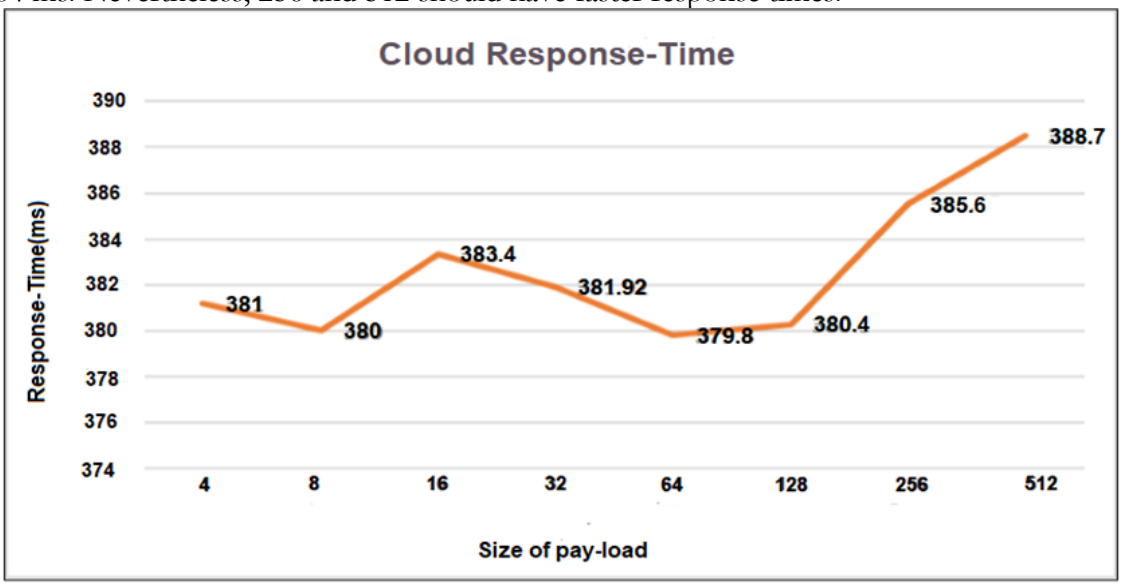

Figure 12. Single user based Response time and Pay-load in Cloud

Figure 13 shows the response time on the y-axis and the number of requests made by users on the x-axis. The outcome shows that response times for scenarios involving more than

40 users rise as the number of users making simultaneous queries increases. Moreover, it has been noted that as payload size grows, in comparison to response time [23].

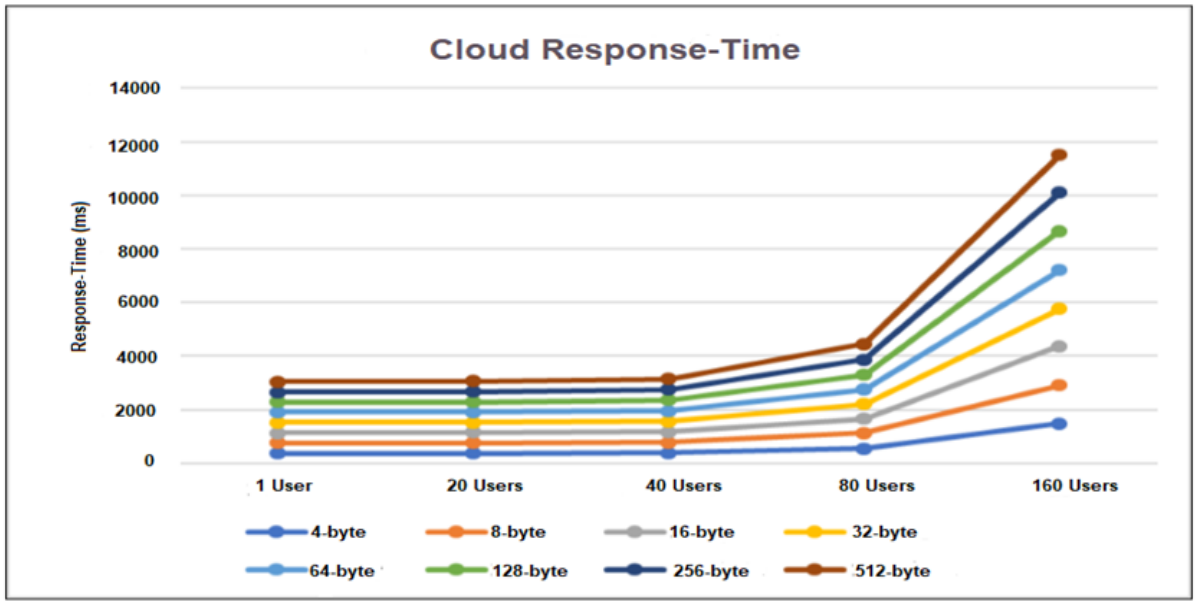

Figure 13. Multi-user based Response time and Pay-load in Cloud

Figure 14 shows the throughput (total number of transactions per second) on the y-axis and the number of users who submit requests, 1, 20, 40, 80, and 160, on the x-axis. We concluded from the findings that an increase in reaction time reduces the number of transactions per second (throughput). Throughput is also reduced when the number of users exceeds 160.

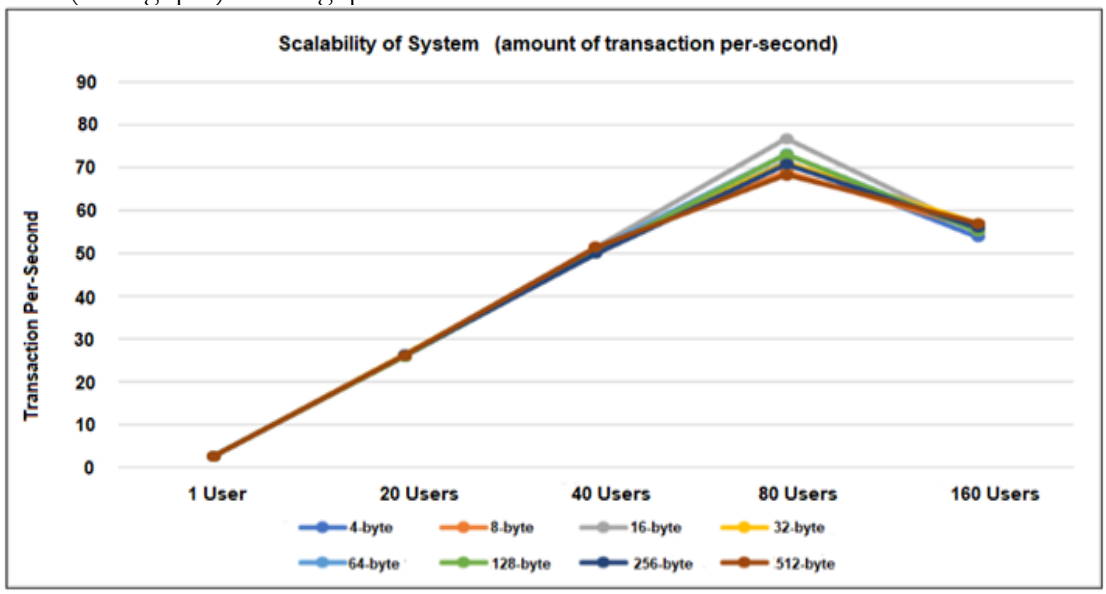

Figure 14. Scalability of System

As depicted in the aforementioned pictures, the designed architecture employs blockchain to provide TDA in IoT by evaluating single and many users. This makes every action in the IoT network responsible. In contrast to earlier works where the centralized and decentralized approaches were applied but all actions were not accountable, the IoT network is, therefore, more secure and accountable with this work.

Conclusions.

In summary, IoT links devices to the Internet to gather and process data in order to supply new services and build an intelligent society. IoT has gained popularity in recent years as a result of its implementation in smart cities and households all around the world. IoT devices use the inefficiently little network, storage, and computing power. So, compared to other devices like mobile phones or PCs, they are most prone to be attacked. The more IoT devices or items that link to the network are used, the more data is collected from smart instruments, and this raises concerns about data privacy and traceability inside the IoT network. Due to its properties, blockchain is a decentralized, immutable database that is documented to support and suitable for TDA. Blockchain offers the primary benefit of an append-only database, which implies that records cannot be modified and are preserved in chronological order. In this work, we have devised a blockchain-based architecture to support TDA in the IoT. This makes every action in the IoT network responsible. So, compared to earlier networks, the IoT network is more trustworthy and secure. Future Projects: Blockchain is employed in this work to enable TDA. Several aspects of the suggested system design have also been explored. Unfortunately, the system still has to be improved, and the currently suggested approach only supports one User. For the next projects, we are optimistic that we will be able to forecast how the suggested design will change as the number of end users grows.

Research's Effects

Organizational and other IoT-based systems are made more secure by the traceability and data privacy-based solution as compared to other IoT systems that have been previously established. Organizational data is more significant than anything else since it serves as the foundation for decision-making and determining the future course of the company. Organizations can secure their data from unauthorized parties by utilizing the proposed solution.

References

[1] M. Banerjee, J. Lee, and K. K. R. Choo, "A blockchain future for internet of things security: a position paper," *Digit. Commun. Networks*, vol. 4, no. 3, pp. 149–160, 2018, doi: 10.1016/j.dcan.2017.10.006.

[2] Y. Zhang, Y. Liu, Z. Jiong, X. Zhang, B. Li, and E. Chen, "Development and assessment of blockchain-IoT-based traceability system for frozen aquatic product," *J. Food Process Eng.*, vol. 44, no. 5, pp. 1–14, 2021, doi: 10.1111/jfpe.13669.

[3] M. A. Khan and K. Salah, "IoT security: Review, blockchain solutions, and open challenges," *Futur. Gener. Comput. Syst.*, vol. 82, pp. 395–411, 2018, doi: 10.1016/j.future.2017.11.022.

[4] Y. P. Tsang, K. L. Choy, C. H. Wu, G. T. S. Ho, and H. Y. Lam, "Blockchain-Driven IoT for Food Traceability with an Integrated Consensus Mechanism," *IEEE Access*, vol. 7, pp. 129000–129017, 2019, doi: 10.1109/ACCESS.2019.2940227.

[5] M. P. Caro, M. S. Ali, M. Vecchio, and R. Giaffreda, "Blockchain-based traceability in Agri-Food supply chain bb," *2018 IoT Vert. Top. Summit Agric. - Tuscany, IOT Tuscany 2018*, pp. 1–4, 2018.

[6] J. Lin, A. Zhang, Z. Shen, and Y. Chai, "Blockchain and IoT based food traceability for smart agriculture," *ACM Int. Conf. Proceeding Ser.*, pp. 1–6, 2018, doi: 10.1145/3126973.3126980.

[7] H. Cui, Z. Chen, Y. Xi, H. Chen, and J. Hao, "IoT data management and lineage traceability: A blockchain-based solution," *2019 IEEE/CIC Int. Conf. Commun. Work. China, ICCC Work. 2019*, pp. 239–244, 2019, doi: 10.1109/ICCChinaW.2019.8849969.

[8] A. Sultan, M. A. Mushtaq, and M. Abubakar, "IoT security issues via blockchain: A review paper," *ACM Int. Conf. Proceeding Ser.*, vol. Part F1481, pp. 60–65, 2019, doi: 10.1145/3320154.3320163.

[9] S. A. H. and E. H. F. Hussain, R. Hussain, "Machine Learning in IoT Security: Current Solutions and Future Challenges," *IEEE Commun. Surv. Tutorials*, doi: 10.1109/COMST.2020.2986444.

[10] A. Sultan, M. S. Arshad Malik, and A. Mushtaq, "Internet of Things Security Issues and Their Solutions with Blockchain Technology Characteristics: A Systematic Literature Review," *Am. J. Comput. Sci. Inf. Technol.*, vol. 06, no. 03, pp. 1–5, 2018, doi: 10.21767/2349-3917.100027.

[11] M. A. Khan, K. Salah B A Bahauddin, Z. U. Multan, and P. B. Khalifa, "IoT security: Review, blockchain solutions, and open challenges," *Futur. Gener. Comput. Syst.*, vol. 82, pp. 395–411, 2018, doi: 10.1016/j.future.2017.11.022.

[12] M. S. Ali, K. Dolui, and F. Antonelli, "IoT data privacy via blockchains and IPFS," *ACM Int. Conf. Proceeding Ser.*, 2017, doi: 10.1145/3131542.3131563.

[13] and R. D. S. Xue, R. K. Lomotey, "Enabling Sensor Data Exchanges in Unstable Mobile Architectures," *IEEE 3rd Int. Conf. Mob. Serv. MS*, pp. 391–398, 2015.

[14] Z. Shelby, K. Hartke, and C. Bormann, "The Constrained Application Protocol (CoAP)," *Rfc 7252*, p. 112, 2014, [Online]. Available: https://www.rfc-editor.org/rfc/pdfrfc/rfc7252.txt.pdf

[15] R. T. Fielding, "Architectural Styles and the Design of Network-based Software Architectures," *Building*, vol. 54, 2000.

[16] D. Wu, S. Si, S. Wu, and R. Wang, "Dynamic trust relationships aware data privacy protection in mobile crowd-sensing," *IEEE Internet Things J.*, vol. 5, no. 4, pp. 2958–2970, 2018, doi: 10.1109/JIOT.2017.2768073.

[17] M. Yevdokymenko, "An adaptive algorithm for detecting and preventing attacks in telecommunication networks," *2016 3rd Int. Sci. Conf. Probl. Infocommunications Sci. Technol. PIC S T 2016 - Proc.*, pp. 175–177, 2017, doi: 10.1109/INFOCOMMST.2016.7905373.

[18] F. Abazari, A. Madani, and H. Gharaee, "Optimal response to computer network threats," *2016 8th Int. Symp. Telecommun. IST 2016*, pp. 729–734, 2017, doi: 10.1109/ISTEL.2016.7881919.

[19] G. Kalnoor and J. Agarkhed, "Pattern matching intrusion detection technique for Wireless Sensor Networks," *Proceeding IEEE - 2nd Int. Conf. Adv. Electr. Electron. Information, Commun. Bio-Informatics, IEEE - AEEICB 2016*, pp. 724–728, 2016, doi: 10.1109/AEEICB.2016.7538389.

[20] A. Dorri, M. Steger, S. S. Kanhere, and R. Jurdak, "A blockchain-based solution to automotive security and privacy," *Blockchain Distrib. Syst. Secur.*, no. December, pp. 95–115, 2019, doi: 10.1002/9781119519621.ch5.

[21] F. Wang, L. Hu, J. Zhou, and K. Zhao, "A survey from the perspective of evolutionary process in the internet of things," *Int. J. Distrib. Sens. Networks*, vol. 2015, 2015, doi: 10.1155/2015/462752.

[22] E. Kaku, R. K. Lomotey, and R. Deters, "Using Provenance and CoAP to track Requests/Responses in IoT," *Procedia Comput. Sci.*, vol. 94, no. MobiSPC, pp. 144–151, 2016, doi: 10.1016/j.procs.2016.08.023.

[23] S. B. Vijayaraghavan Varadharajan, "Connectivity Frameworks for Smart Devices, 2016," *ISBN*.