Advancing Secure and Efficient Sensor Data Integration in Smart Cities: A Blockchain-Based Approach

Abdul Rafay Saeed Khan , Hassan Jamil Syed * , Junaid Shuja , Azeem Khan , Faizan Qamar * , Quang Ngoc Nguyen *

Posted Date: 5 June 2024

doi: 10.20944/preprints202406.0230.v1

Keywords: Internet of Things; Cloud Computing; Blockchain; Smart Contract; Sensors; Data Storage

Article

Advancing Secure and Efficient Sensor Data Integration in Smart Cities: A Blockchain-Based Approach

Abdul Rafay Saeed Khan[1], Hassan Jamil Syed[1,2,3]*, Junaid Shuja[4], Azeem Khan[5], Faizan Qamar[6]* and Quang Ngoc Nguyen[7,8]

1 FAST School of Computing, National University of Computer and Emerging Sciences, Karachi 75030, Pakistan; k201337@nu.edu.pk
2 Faculty of Computing and Informatics, Universiti Malaysia Sabah, Jalan UMS, Kota Kinabalu 88400, Sabah, Malaysia
3 Cyber Security Research Lab, Faculty of Computing and Informatics, Universiti Malaysia Sabah, Jalan UMS, Kota Kinabalu 88400, Sabah, Malaysia
4 Department of Computer and Information Sciences, Universiti Teknologi PETRONAS, Seri Iskandar, Malaysia; junaid.shuja@utp.edu.my
5 Faculty of Islamic Technology, University Islam Sultan Sharif Ali, Brunei Darussalam; azeem@unissa.edu.bn
6 Center for Cyber Security, Faculty of Information Science and Technology (FTSM), Universiti Kebangsaan Malaysia (UKM), 43600 UKM Bangi, Selangor, Malaysia
7 Faculty of Science and Engineering, Waseda University, Shinjuku-ku, Tokyo 169-0051, Japan
8 Posts and Telecommunications Institute of Technology, Hanoi, Vietnam
* Correspondence: shjamil@ums.edu.my(H.J.S.); faizanqamar@ukm.edu.my (F.Q.); quang.nguyen@fuji.waseda.jp/nnquang@ptit.edu.vn (Q.N.N.)

Abstract: As technology advances, people are increasingly adopting IoT (Internet of Things) devices in smart cities. These devices are equipped with sensors that are connected everywhere, from energy infrastructure and transportation to telecommunications, health and human services, waste management, public safety, and more. People want to access sensor data anytime and everywhere to make better decisions and improve their lives. To achieve this, a sensor integration model has been developed where multiple sensors are integrated under one gateway to manage and share sensor data. However, effective sharing of sensor data requires data security, privacy, confidentiality, fairness, effective architectural design, and low communication and computation costs. These issues need to be addressed in the sensor integration model in smart cities. Existing storage models face challenges in managing sensor data effectively. To overcome these issues and hurdles, a data storage framework leveraging blockchain technology has been proposed within the sensor integration model. This framework ensures data security, privacy, confidentiality, and a fair and transparent financial model, all while minimizing communication and computation costs.

Keywords: Internet of Things; Cloud Computing; Blockchain; Smart Contract; Sensors; Data Storage

1. Introduction

As technology advances and urban populations grow, cities face many challenges spanning waste management, healthcare, water distribution, and education. In response, smart cities have emerged as a promising solution, leveraging cutting-edge technologies to tackle these issues effectively. Central to the concept of smart cities is integrating sensors into Internet of Things (IoT) devices deployed across various urban domains such as hospitals, roads, homes, and public spaces [1]. This integration allows for collecting and utilizing sensor data to enhance quality of life and operational efficiency [2].

Moreover, recent technological advancements have propelled smart cities towards greater sophistication by integrating electronic and physical sensors into urban infrastructure. This integration, coupled with the widespread adoption of mobile devices, underscores the importance of exploring various concepts within the smart city framework, including motivations, object aspects, contributors, and legal frameworks [3]. As urban populations continue to grow, projections indicate a significant increase in networked sensors by 2030, highlighting the urgent need for robust sensor and data management strategies [4].

 doi:10.20944/preprints202406.0230.v1

The Sensor Integration Model, comprising sensor service providers, data providers, and data purchasers, offers a structured framework for streamlined sensor deployment and data access within smart cities [5]. Despite its potential benefits, unresolved issues persist across economic, social, and technical domains. This study aims to address financial and technological concerns surrounding data storage, accuracy, confidentiality, security, and fairness within the Sensor Integration Model [6]

In particular, achieving fairness in data transactions necessitates the development of equitable business models and trust mechanisms. Blockchain technology is a promising solution to enhance fairness and transparency within the Sensor Integration Model. [7]. However, challenges such as computing costs and scalability hinder the widespread adoption of blockchain-based solutions in smart cities, emphasizing the need for continued research and innovation in this field [8,9]

The proposed solution introduces a comprehensive sensor integration model and data storage framework comprising four main participants: sensors and sensor owners, sensor data consumers, sensor service providers, and blockchain technology. Sensors deployed to detect various physical occurrences are associated with sensor owners who may choose to offer their data for free or for a fee [10,11]. The sensor service provider acts as a platform facilitating registration, data transactions, and storage, leveraging blockchain for secure and transparent operations. Sensor data consumers, ranging from governmental bodies to commercial entities, access sensor data via the platform, with transactions executed seamlessly through blockchain smart contracts. The security requirements of the data storage framework include pseudonymity, unlinkability, and traceability, ensuring confidentiality and authenticity in data transactions. Infrastructure development involves sensor integration, registration processes, secure data storage mechanisms, and implementation of blockchain technology [12]. Testing and optimizing the proposed system, including gas cost analysis in Ethereum, ensure efficiency and scalability in real-world applications.

The subsequent sections of this paper are structured as follows: Section 2 delves into the exploration of related works. Section 3 outlines the architecture of the proposed sensor integration model and data storage framework. Section 4 details the implementation stages of the proposed solution. Section 5 evaluates the proposed solution based on the obtained results. Finally, Section 6 concludes the paper by summarizing the findings.

2. Literature Review

The sensor data storage framework uses cloud services to interact with the IoT. It provides the data effectively and securely based on the user's requirements. The Internet of Things (IoT) allows for the virtual communication of objects and the physical communication, sensing, computation, and processing of data via sensors and other physical devices [13]. The applications that belong to the IoT usually communicate with the services of physical devices or sensors [14]. Sensors on smart devices perform the operations related to sensing data for an IoT platform. The cloud servers manage sensor data collection, and all the required data is obtained through an IoT application in a pay-as-you-go manner. The quality of predicted data analytics is increased, and seamless internal and external networks support efficient automated mechanisms by checking all the data that are consumed by the sensors on physical devices. The IoT continues to face its most significant challenges related to data management in terms of its analysis, consumption, and usage. IoT must be designed and managed so that it can take advantage of the nearly limitless capabilities that Cloud Computing can offer, for instance, to make up for the smart device's technological gaps [15]. The Cloud servers can act as a transitory layer among IoT Objects and IoT applications to manage resources effectively.

 doi:10.20944/preprints202406.0230.v1

Table 1. Summary of Literature review

Reference	Focus	Method/Approach	Advantages	Limitations
[13]	Integration of IoT with Cloud Computing	Utilization of Cloud services for secure and effective data interaction; improving data analytics quality; managing resources effectively.	Improved data analytics; Efficient resource management; Secure data interaction.	Dependency on cloud infrastructure; Potential privacy concerns.
[14]	IoT Applications & Communication	IoT applications communicate with physical devices or sensors; sensors perform data sensing; cloud servers manage data collection and analytics.	Efficient IoT communication; Centralized data management; Scalability.	Reliance on cloud connectivity; Security vulnerabilities in data transfer.
[15]	Access Control in Cloud Servers	Integration of cryptographic techniques and role-based access management for safe information storage and exchange in cloud servers.	Enhanced data security; Controlled access to information; Secure data exchange.	Complexity of cryptographic implementations; Key management challenges.
[16]	Attribute-Based Cloud Data Sharing System	Proposal of an attribute-based cloud data-sharing system for low-resource mobile users, employing offline partial encryption and system characteristics.	Reduced computational overhead; Secure data sharing; Scalability.	Complexity in attribute management; Potential key exposure risks.
[17]	Ciphertext Policy Attribute-Based Encryption	Proposal of a method for cloud storage with effective user revocation, low computing costs, and collusion attack resistance.	Efficient user revocation; Flexible access control; Resistance to collusion attacks.	Increased computational overhead; Complexity in policy management.
[18]	Secure Proxy Re-encryption Method for IoT Data	Introduction of a secure proxy re-encryption method for cloud-based systems managing IoT data, ensuring bidirectional ciphertext conversions.	Bidirectional data conversion; Efficient data transfer; Enhanced data security.	Computational overhead in re-encryption; Potential performance impact.
[19]	IoT Access Control System based on Blockchain	Integration of blockchain, attribute-based access control, and identity-based signatures for enhanced IoT access control and security.	Decentralized access control; Immutable access policies; Enhanced security.	Blockchain scalability issues; Increased network latency.
[20]	Security in Sensing as a Service (SaaS) IoT	Proposal of a secure authentication and key management system for SaaS IoT, eliminating traditional smart cards and ensuring efficient data access.	User-friendly authentication; Efficient data access; Enhanced security measures.	Dependency on centralized authentication; Potential single point of failure.
[21]	Decentralized Reputation System for Crowd-Sensing	Introduction of a decentralized reputation and reward system for mobile crowd-sensing networks, enhancing involvement and addressing privacy concerns.	Increased user participation; Improved data credibility; Privacy protection.	Scalability challenges in reputation management; Potential manipulation of reputation scores.
[22]	Energy Optimization in Wireless Sensor Networks	Proposal of a grouping memetic algorithm for wireless sensor network energy optimization, targeting IoT applications in smart cities.	Energy-efficient sensing; Enhanced network coverage; Effective task planning.	Computational complexity; Algorithm parameter tuning requirements.
[23]	Blockchain-based Solutions for IoT Challenges	Advocacy for blockchain-based remedies to IoT challenges, addressing issues like security, privacy, resource utilization, and interoperability.	Improved security and privacy; Decentralized governance; Enhanced interoperability.	Blockchain scalability issues; Increased computational overhead.

The authorization of access control standards has been aided by integrating cryptographic techniques and role-based access management in cloud servers, ensuring safe information storage and

 doi:10.20944/preprints202406.0230.v1

exchange [16]. Customers can scramble their data, allowing only those who are officially authorized to decipher and access it per defined access agreements. However, despite these precautions, confidence in the effective management of customer data by access control providers still exists, underscoring the growing difficulty of ensuring reliability in cloud environments. In another work in [17], the authors propose a mechanism called Role-Based Get to Control (RBAC), that is introduced for ensuring dependability and improving information protection and security in cloud servers. Their program enables data owners to assess the dependability of access configurations and components, assisting in decisions about storing disorganized data. They provide a dependable cloud capacity framework engineering by combining a reliability demonstration with cryptographic RBAC arrangements, reducing risks and advancing information owner evaluations. Moreover, the authors in [18] suggest an attribute-based cloud data-sharing system for low-resource mobile users. The approach minimizes computational work by using offline partial encryption and system characteristics. It uses Chameleon hashing to generate cipher-texts instantaneously, disguising offline ciphertexts with online ones. The technology ensures safe data exchange in cloud contexts by effectively thwarting attacks via adaptively determined ciphertexts.

For cloud storage, the study in [19] proposed ciphertext policy attribute-based encryption, incorporating effective user revocation. The method keeps costs low by contracting with cloud providers to do sophisticated computations, even if it has high computational expenses. It features minimal and stable computing costs on local devices, is immune to collusion attacks, and guarantees file content and key confidentiality. Hence, it is appropriate for devices with constrained resources. Another work by [20] focuses on a secure proxy re-encryption method for cloud-based systems that manage IoT data that is outsourced from IoT devices. Their method is predicated on the notion that bilinear inverse Diffie-Hellman is hard. Their method enables bidirectional ciphertext conversions even in scenarios involving numerous IoT nodes. Their system's re-encryption processes only take one exponentiation, or 54 milliseconds, to complete while sending data between 100 nodes. In contrast, the study in [21] introduced an IoT access control system based on blockchain. Blockchain, attribute-based access control, and identity-based signatures are all integrated into one system. They created distinct blockchain ledgers for each functional area into which they separated the IoT platform. Access policies, digests, and attributes are managed using these ledgers. Their approach combines Identity-based signature (IBS) and Hyperledger Fabric (HLF) channel architecture to prevent DDoS assaults and to identify decentralized policy decision points for runtime policy enforcement.

Another work in [22] seeks to resolve security issues with the Sensing as a Service (SaaS) IoT approach in their study. In this paradigm, sensor data is sold on a marketplace. The researchers suggest a secure authentication and key management system to reduce vulnerabilities in the open service industry. Their approach does away with traditional smart cards by providing a user-friendly website that allows efficient and safe access to sensor data. A publish/subscribe approach powers the system, making it simple for users to register, log in, and request data. The data is sent to the relevant sensor and fog nodes upon receipt of a request. The system verifies user identity and data integrity before accessing the requested data. Furthermore, a decentralized reputation and reward system for mobile crowd-sensing networks has been proposed by [23]. Their objectives are to increase involvement and address weaknesses like intrusions and invasions of privacy. They use reputation management strategies and cutting-edge encryption standards to do this. Similarly, the study in[24] proposes a grouping memetic algorithm for wireless sensor network energy optimization. This algorithm is designed especially for the use of the Internet of Things (IoT) in smart cities. The algorithm effectively plans out sensing tasks by solving the SET K-COVER issue. The ultimate objective is reducing energy consumption and maximizing coverage among heterogeneous sensor nodes.

Moreover, the authors in [25] offer a thorough analysis of the difficulties encountered in the Internet of Things space and suggest a blockchain-based remedy. In order to address issues like the removal of centralized authority, faster peer messaging, efficient resource utilization, secure code deployment, data security, transparency, improved interoperability, identity management, and

 doi:10.20944/preprints202406.0230.v1

reliability improvement, they advocate for adopting an integrated architectural design incorporating blockchain technology. Sensing, cloudlet, and dew layers make up the suggested architecture. To address issues with security, privacy, device constraints, massive data processing, and network infrastructure traffic, blockchain is integrated into the dew and cloudlet levels. Smart contracts are used in various IoT systems to facilitate protocol integration and authentication, reducing security risks, data storage requirements, computational intensity, network congestion, device limits, and data privacy.

Table 1 summarizes the related work discussed above and outlines each study's focus, method/approach, advantages, and limitations.

3. Proposed Solution

3.1. System Model

The proposed sensor integration model and data storage framework involve four main participants: sensors and sensor owners, sensor data consumers, sensor service providers, and blockchain technology. Figure 1 illustrates the communication flow of the Sensor Integration Model. Further details about their roles are provided below.

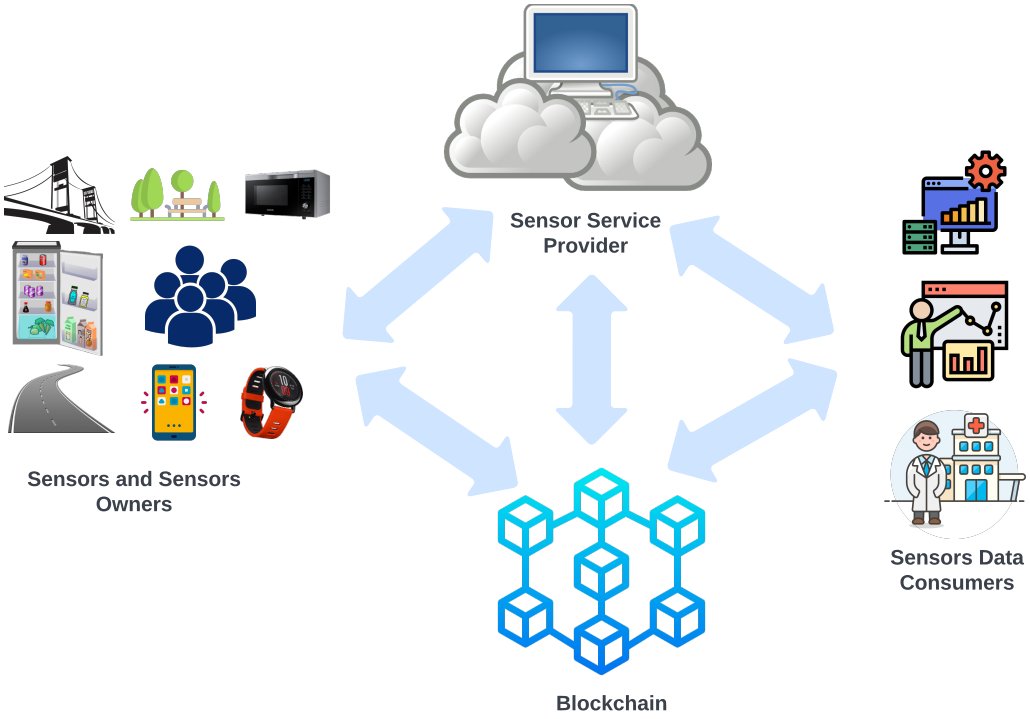

Figure 1. Sensor Integration Model communication Flow

3.2. Sensors and Sensor Owners

Typically, sensors are connected to items to detect, measure, or sense a physical occurrence, like the temperature of the environment, relative humidity, moisture level, etc. For example, streets and roads contain sensors that monitor traffic and atmospheric conditions, and microwaves or coffee makers might also have sensors that identify activities like the daily consumption count. Certain anti-disassembled mechanisms are presumed to protect the sensors from being disassembled. Additionally, each sensor is assumed to possess security features for securely storing private keys and managing cryptographic operations. The data collected from deployed sensors can be used to evaluate conditions of different requirements more thoroughly or assist us in quickly identifying user choices and actions. Every sensor often relates to a specific person, referred to as the sensor owner, and that person's association with the registered sensors can shift from time to time. The sensor owner might offer the

 doi:10.20944/preprints202406.0230.v1

sensor data for free without any fee or set a price to recover costs or enhance the user experience. Figure 2 depicts the flow of sensor data provider registration.

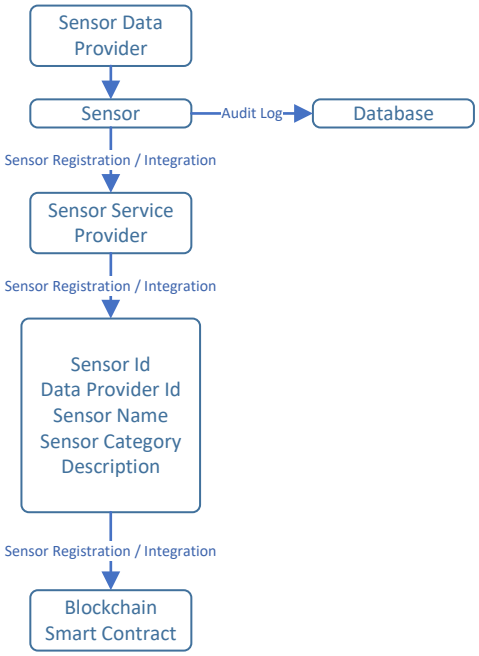

Figure 2. Sensor data provider registration flow

3.2.1. Sensor Service Provider

Even though suggested by its name, this entity is in charge of offering specialized services. Initially, it provides a platform for registering such entities, like sensor owners and sensor data consumers [26]. Those individuals that are authorized and registered on the platform are able to purchase or sell their sensor data or get additional services offered on the platform by the sensor service provider. Those features involve finding the targeted sensor's owner or sensor data consumer, cloud storage is used for safely maintaining the sensor data, tracking accountability for handling disputes, and more [27]. This is important to note that registrations and finding functions are carried through a blockchain ledger, or more accurately, a smart contract. The assumption is made that the sensor service provider may be trustworthy, yet there is uncertainty regarding its dedication to rigorously adhere to responsibilities such as registration, findings, and data storage. Figure 3 illustrates the flow of uploading sensor data.

 doi:10.20944/preprints202406.0230.v1

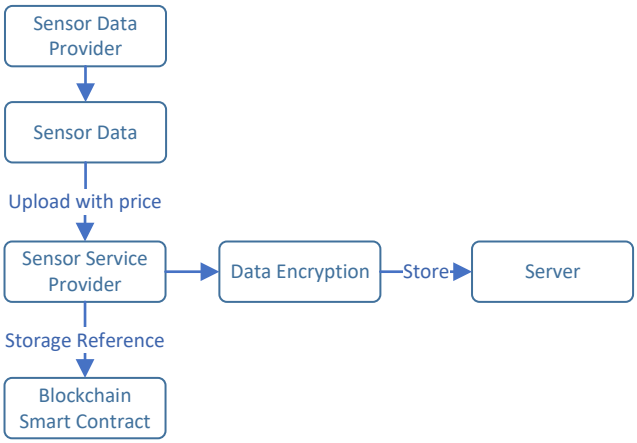

Figure 3. Upload Sensor Data Flow

3.2.2. Sensor Data Consumers

Those members could be governmental bodies, commercial entities, schools, universities, or science and research societies that intend to buy the sensor data for the abovementioned purposes. Owners of the sensors can decide on a recurring payment schedule, a cost-per-consumption model, or even opt to give away the sensor data for free. However, these possibilities are not considered, and instead, a consistent periodic payment strategy is selected. Each individual must sign up individually and request legitimate authorization from the sensor service provider. After that, they can use the blockchain to look up the desired sensor's owner and buy the sensor data. According to our idea, blockchain smart contracts will perform the transaction procedure seamlessly. Figures 4, 5 and 6 provide a visual representation of the different flows in the system. Figure 4 shows the registration flow for the sensor data consumer, Figure 5 illustrates the search for sensor data flow, and Figure 6 depicts the complete transaction flow.

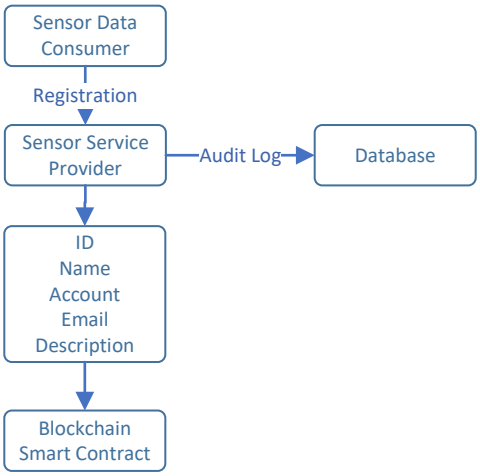

Figure 4. Sensor data consumer registration flow

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 June 2024 doi:10.20944/preprints202406.0230.v1

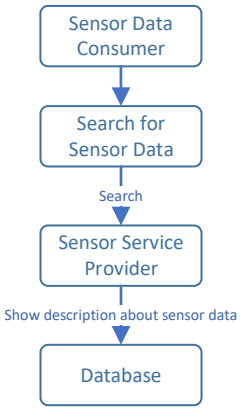

Figure 5. Search for sensor data flow

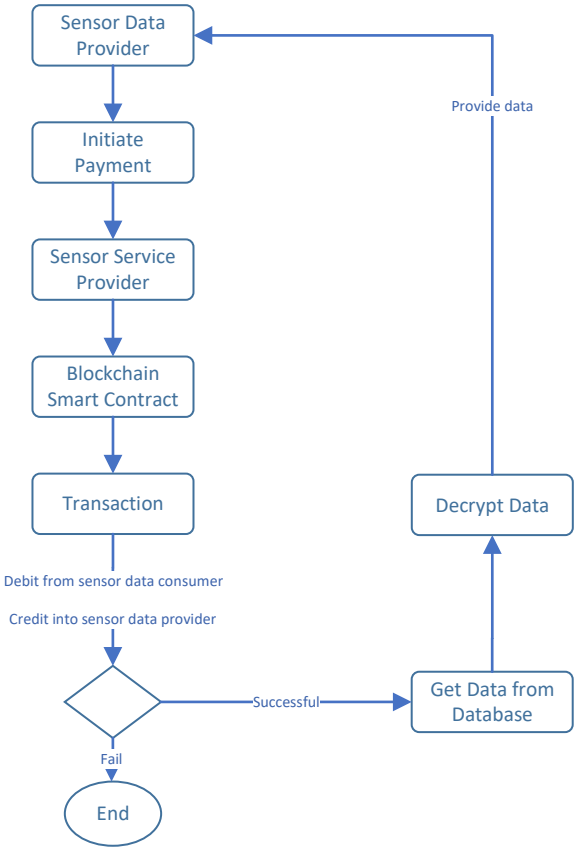

Figure 6. Transaction flow

3.2.3. Blockchain

Any blockchain that supports the implementation of smart contracts, such as Ethereum or Hyperledger, could be implemented into our system [28]. Smart contracts serve two significant purposes. Initially, it will act as a registration mechanism for entities permitted by the sensor service provider and keep track of their pseudonym identification. Secondly, it can mediate between the sensor owner and the sensor data customers, deciding whether or not to transmit the paid amount to

 doi:10.20944/preprints202406.0230.v1

the sensor owner. Every time a smart contract is engaged, the transaction would specifically be issued. The detailed set of Algorithms (Algorithm 1 to Algorithm 9) are explained below.

Algorithm 1 Sensor Data Consumer Initialization

Data Structures: - *DataConsumersIds* : String array (Stores sensor data consumer IDs) List of all consumer identifiers - *SensorDataConsumer* : Struct (Defines sensor data consumer model) - *DCid* : String (Unique identifier for the consumer) - *account* : String (Blockchain account address) - *Name* : String (Name of the sensor data consumer) - *Country* : String (Country of the sensor data consumer) - *City* : String (City of the sensor data consumer) - *Email* : String (Email address of the consumer) - *OrganizationName* : String (Name of the organization) - *Description* : String (Optional description of the consumer) - *isActive* : Boolean (Flag indicating active state) - *DataConsumers* : Mapping (String to *SensorDataConsumer*) Dictionary for storing consumer data by ID

Algorithm 2 RegisterSensorDataConsumer

Input:
- *DCid* : String (Unique identifier for the sensor data consumer)
- *account* : Address (Blockchain account address)
- *Name* : String (Name of the sensor data consumer)
- *Country* : String (Country of the sensor data consumer)
- *City* : String (City of the sensor data consumer)
- *Email* : String (Email address of the consumer)
- *OrganizationName* : String (Name of the organization)
- *Description* : String (Optional description of the consumer)
- *isActive* : Boolean (Flag indicating active state)

Precondition:
- $DataConsumers[DCid].isActive \neq$ false {Consumer must not already be active}

Procedure:
Update Consumer List:
$DataConsumersIds \leftarrow \text{Push}(DCid)$ {Add DCid to list of consumer IDs}
Create Consumer Entry:
$DataConsumers[DCid] \leftarrow \text{SensorDataConsumer}(DCid, account, Name, Country, City, Email, OrganizationName, Description, isActive)$ {Create new consumer object and store in mapping}

3.3. Security Requirements for the Data Storage Framework System

This section outlines essential measures to ensure sensor data's integrity, confidentiality, and accountability. This encompasses pseudonym management, unlinkability, traceability, revocation mechanisms, authenticity, confidentiality, and fairness, as detailed in subsequent subsections. These requirements form the foundation for the secure operation of the data storage framework, safeguarding sensitive information and promoting trust in the system's functionality and governance.

3.3.1. Pseudonyms

The sensor service provider can register several different usernames for both sensor data consumers and sensor data providers [29]. This characteristic indicates a single person can access multiple pseudonyms to use diverse sensor services. Therefore, the objective of protecting identification and confidentiality is achieved; in other words, the actual identity is concealed behind the pseudonyms.

3.3.2. Being Unlinkable and Tracability

Any pseudonym being used various sensor services registered with the same real name are not supposed to be linked to one another. But, the sensor service provider must be capable of disclosing the real identity of such a particular pseudonym to ensure responsibility for such untruthful or even harmful acts.

 doi:10.20944/preprints202406.0230.v1

Algorithm 3 GetSensorDataConsumerDetails

Input:
- *DCid* : String (Unique identifier for the sensor data consumer)

Output:
A tuple containing:
- *account* : String (Account associated with the provider)
- *Name* : String (Name of the sensor data consumer)
- *Country* : String (Country of the sensor data consumer)
- *City* : String (City of the sensor data consumer)
- *Email* : String (Email address of the consumer)
- *OrganizationName* : String (Name of the organization)
- *isActive* : Boolean (Flag indicating active state)

Procedure:
Lookup Data Provider:
Access data store named 'DataConsumers'.
Search for entry with key *DCid* in 'DataConsumers'.
Check for Existence:

if $DataConsumers[DCid]$ is undefined **then**
    Return "Sensor data consumer not found" {Error message}
end if
Extract Details:
Extract details from 'DataConsumers[DCid]':
$account \leftarrow DataConsumers[DCid].account$
$Name \leftarrow DataConsumers[DCid].Name$
$Country \leftarrow DataConsumers[DCid].Country$
$City \leftarrow DataConsumers[DCid].City$
$Email \leftarrow DataConsumers[DCid].Email$
$OrganizationName \leftarrow DataConsumers[DCid].OrganizationName$
$isActive \leftarrow DataConsumers[DCid].isActive$
Return Data:
Return $(account, Name, Country, City, Email, OrganizationName, isActive)$

Algorithm 4 Sensor Data Provider Initialization

Data Structures:
- *DataProvidersIds* : String array (Stores sensor data provider IDs) {List of all provider identifiers}
- *SensorDataProvider* : Struct (Defines sensor data provider model)
  - *DPid* : String (Unique identifier for the provider)
  - *account* : String (Blockchain account address)
  - *Name* : String (Name of the sensor data provider)
  - *Country* : String (Country of the sensor data provider)
  - *City* : String (City of the sensor data provider)
  - *Email* : String (Email address of the provider)
  - *Description* : String (Optional description of the provider)
  - *isActive* : Boolean (Flag indicating active state)
- *DataProviders* : Mapping (String to *SensorDataProvider*) {Dictionary for storing provider data by ID}

 doi:10.20944/preprints202406.0230.v1

Algorithm 5 RegisterSensorDataProvider

Input:
- *DPid* : String (Unique identifier for the sensor data provider)
- *account* : Address (Blockchain account address)
- *Name* : String (Name of the sensor data provider)
- *Country* : String (Country of the sensor data provider)
- *City* : String (City of the sensor data provider)
- *Email* : String (Email address of the provider)
- *Description* : String (Optional description of the provider)
- *isActive* : Boolean (Flag indicating active state)

Precondition:
$DataProviders[DPid].isActive \neq$ true {Provider must not already be active}
Procedure:
Update Provider List:
$DataProvidersIds \leftarrow Push(DPid)$ {Add DPid to list of provider IDs}
Create Provider Entry:
$DataProviders[DPid] \leftarrow SensorDataProvider(DPid, account, Name,$
Country, City, Email, Description, isActive) {Create new provider object and store in mapping}

Algorithm 6 GetSensorDataProvidersDetails

Input:
- *DPid* : String (Unique identifier for the sensor data provider)

Output:
A tuple containing:
- *account* : String (Account associated with the provider)
- *Name* : String (Name of the sensor data provider)
- *Country* : String (Country of the sensor data provider)
- *City* : String (City of the sensor data provider)
- *Email* : String (Email address of the provider)

Procedure:
Lookup Data Provider:
Access data store named 'DataProviders'.
Search for entry with key *DPid* in 'DataProviders'.
Check for Existence:

if $DataProviders[DPid]$ is undefined **then**
Return "Sensor data provider not found" {Error message}

end if
Extract Details:
Extract details from 'DataProviders[DPid]':
$account \leftarrow DataProviders[DPid].account$
$Name \leftarrow DataProviders[DPid].Name$
$Country \leftarrow DataProviders[DPid].Country$
$City \leftarrow DataProviders[DPid].City$
$Email \leftarrow DataProviders[DPid].Email$
$isActive \leftarrow DataProviders[DPid].isActive$
Return Data:
Return $(account, Name, Country, City, Email, isActive)$

 doi:10.20944/preprints202406.0230.v1

Algorithm 7 Sensor Integration Initialization

Data Structures:
- *SensorsIds* : String array (Stores sensor IDs) {List of all sensor identifiers}
- *Sensors* : Struct (Defines sensor model)
  - *Sid* : String (Unique identifier for the sensor)
  - *DPid* : String (Sensor data provider ID) {ID of the associated provider}
  - *SensorName* : String (Name of the sensor)
  - *SensorCategory* : String (Category of the sensor)
  - *Description* : String (Optional description of the sensor)
  - *isActive* : Boolean (Flag indicating active state)
- *SensorsReg* : Mapping (String to *Sensors*) {Dictionary for storing sensor data by ID}

Algorithm 8 RegisterSensor

Input:
- *Sid* : String (Unique identifier for the sensor)
- *DPid* : String (Sensor data provider ID, associated with the sensor)
- *SensorName* : String (Name of the sensor)
- *SensorCategory* : String (Category of the sensor)
- *Description* : String (Optional description of the sensor)
- *isActive* : Boolean (Flag indicating active state)

Precondition:
$SensorsReg[Sid].isActive \neq$ true {Sensor must not already be active}

Procedure:
Update Sensor List:
$SensorsIds \leftarrow Push(SensorAndDataProviderId(\text{Sid}, DPid))$ {Combine Sid and DPid, then push to list}
Create Sensor Entry:
$SensorsReg[Sid] \leftarrow Sensors(Sid, DPid, SensorName, SensorCategory,$
Description, isActive) {Create new sensor object and store in mapping}

 doi:10.20944/preprints202406.0230.v1

Algorithm 9 GetSensorsDetails

Input:
- *Sid* : String (Unique identifier for the sensor)

Output:
A tuple containing:
- *Sid* : String (Unique identifier for the sensor)
- *DPid* : String (Sensor data provider ID, associated with the sensor) {Assuming DPid exists in Sensors struct}
- *SensorName* : String (Name of the sensor)
- *SensorCategory* : String (Category of the sensor)
- *Description* : String (Optional description of the sensor)
- *isActive* : Boolean (Flag indicating active state)

Procedure:
Lookup Sensor:
Access data store named 'SensorsReg'.
Search for entry with key *Sid* in 'SensorsReg'.
Check for Existence:

if $SensorsReg[Sid]$ is undefined **then**
    Return "Sensor not found" {Error message}

end if
Extract Details:
Extract details from 'SensorsReg[Sid]':
$sid \leftarrow SensorsReg[Sid].Sid$
$DPid \leftarrow SensorsReg[Sid].DPid$
$SensorName \leftarrow SensorsReg[Sid].SensorName$
$SensorCategory \leftarrow SensorsReg[Sid].SensorCategory$
$Description \leftarrow SensorsReg[Sid].Description$
$isActive \leftarrow SensorsReg[Sid].isActive$
Return Data:
Return $(sid, DPid, SensorName, SensorCategory, Description, isActive)$

 doi:10.20944/preprints202406.0230.v1

3.3.3. A Successful Revocation

The platform must additionally include a reliable revocation method that can be used to revoke the authorization for all pseudonyms that belong to the fraudulent entity. Everyone who engages in misleading or malicious behavior would not have the ability to access sensor services and may potentially face punishment.

3.3.4. Authenticity

While transmitting data, the sensing service and sensor must be capable of fully identifying one another, particularly whenever the sensor service provider could verify that perhaps the sensor data received is indeed coming from the actual sensor [30].

3.3.5. Confidentiality

The sensor data must be safely sent and kept because the sensor service provider is trustworthy but interested. Additionally, the operator of the sensor, as well as the data consumer, must covertly exchange the key necessary to decrypt the encrypted sensor data.

3.3.6. Fairness

In the proposed system, fairness must be realized even without the involvement of an additional party. A malicious sensor data consumer cannot get the genuine data of the sensor without paying a fair service cost. If the sensor owner gives the sensor data to consumers with inaccurate sensor information, they are punished.

3.4. Infrastructure

First, we gather sample data of multiple heterogeneous sensors that are used in smart cities then we work on the integration and registration process of sensors on the sensor integration model service, which involves registering all of the information about the sensor and its sensing data. Next, we'll focus on the sensor data consumer registration process on the sensor integration model service. After that, other features offered by the sensor integration model service will be developed, including finding sensor data owners and consumers and a secure storage framework of sensor data. Our data storage framework system will be built using the encryption algorithm, a secure symmetrical encryption technique, and a secure signature algorithm. Then, we work on the blockchain for our data storage framework. Any blockchain that supports smart contracts (like Ethereum or Hyperledger) can be used in our proposed method for data storage. A smart contract has two functions. It can first act as a registration mechanism for the entities permitted by the sensor service provider, maintaining their pseudonym information. Second, it can mediate between the sensor owner and the data customer, deciding whether or not to transmit the prepaid deposit to the sensor owner. The remix3 browser solidity, which provides an integrated development environment for testing smart contracts, will be used to construct the smart contract prototype. After installing the smart contract, we will test the gas costs of each function. The cost of gas in the Ethereum system serves as a benchmark for the payment of certain related computations or storage.

Materials and Methods should be described with sufficient details to allow others to replicate and build on published results. Please note that the publication of your manuscript implies that you must make all materials, data, computer code, and protocols associated with the publication available to readers. Please disclose any restrictions on the availability of materials or information at the submission stage. New methods and protocols should be described in detail, while well-established methods can be briefly described and appropriately cited.

Research manuscripts reporting large datasets that are deposited in a publicly available database should specify where the data have been deposited and provide the relevant accession numbers. If the accession numbers have not yet been obtained at the time of submission, please state that they will be provided during review.

 doi:10.20944/preprints202406.0230.v1

Interventional studies involving animals or humans and other studies that require ethical approval must list the authority that provided approval and the corresponding ethical approval code.

4. Implementation and Evaluation

In this section, we present the details of the implementation and experimental setup of the sensor integration model. Additionally presented are the data-gathering techniques and evaluation tools used. The section also addresses the data obtained, evaluation measures, and data-gathering methods to implement the proposed model. The remaining section have been organized as follows: Section 4.1 defines the experimental setup. Section 4.2 defines the implementation of the Sensor integration model. Section 4.3 explains the methods for gathering the data used in the proposed framework. Section 4.4 describes various evaluation metrics that were applied to assess the system's performance.

4.1. Experimental Setup

Figure 7 presents the complete architecture diagram of the Sensor Integration Model. Moreover, The details of the experimental setup are described in this subsection. It defines the environment configurations and deployment details.

4.1.1. Smart Contract

Initially, the prototype for the smart contract is established using a browser-based Solidity development environment, such as Remix3. The core of our simulation is built upon Solidity for creating smart contracts. It gives users access to the Integrated Development Environment (IDE) for deployment and testing of smart contracts. The configurational setup of the smart contract consists of:

- Programming Language: Solidity
- Compiler: Remix IDE - version 0.33.0
- Ethereum virtual machine version: Default version of compiler
- The deployment environment: Ganache
- Cryptocurrency: Ether
- Cryptocurrency wallet: Metamask
- Plugins: Debugging, deploying and running transactions, Solidity statistical analysis, and Solidity testing

Solidity is an object-oriented programming language for building smart contracts on different blockchain systems.

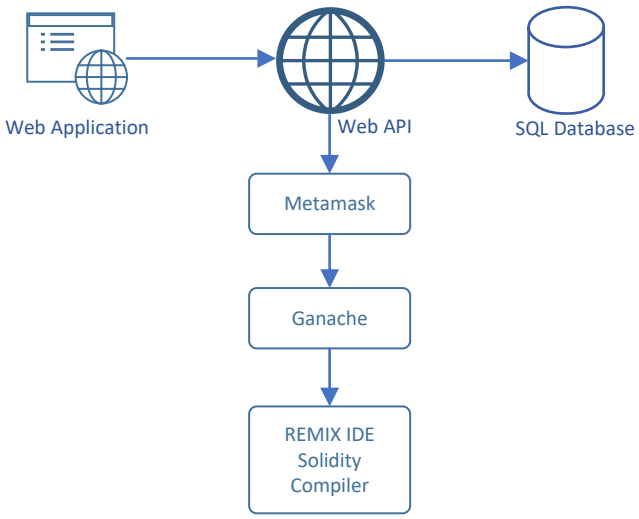

Figure 7. Sensor Integration Model Architecture Diagram

 doi:10.20944/preprints202406.0230.v1

4.1.2. Smart Contract Deployment Environment

Then, we set up the deployment environment of the smart contract. We have deployed the smart contract on Truffle Ganache. For blockchains corresponding with Ethereum and the EVM (Ethereum Virtual Machine), truffle provides a development platform. It functions like a DevOps environment to enable the deployment and testing of smart contracts. Ganache, a component within the "Truffle Suite" of items, works as a local host blockchain where you may securely deploy the smart contracts. Ganache is a private blockchain for developing distributed applications quickly with Filecoin and Ethereum. It may be applied throughout the whole development cycle, allowing you to create, distribute, and evaluate your dApps in a secure and predictable environment. You may develop and test programs using your PC.

4.1.3. Cryptocurrency Wallet - Metamask

We have set up our ganache environment, connected the ganache environment with Metamask, and linked the ganache test accounts on Metamask wallet. MetaMask is used When interacting with the Ethereum blockchain. It enables individuals to communicate with decentralized applications by giving them access to their Ethereum wallet via an extension for their browser. To connect with MetaMask we have provided the Network Name, RPC URL, Chain ID, Currency Symbol, Block Explorer URL, and account private key.

4.1.4. Sensor Integration Model User Interface

To provide the User Interface for our proposed model. A web application has been developed to provide sensor data consumers and sensor data providers access to a management portal for their data. We have developed a web application on .NET core 6.0. .NET is an open-source, cross-platform development environment that can create various applications. Many large-scale programs employ the highly efficient runtime on which.NET is based in their production environments.

4.1.5. Data Management

We have set up the database for the management of the auditing of sensor service providers, sensor data consumers, system-level configurations, and sensor data stored in encrypted form.

4.2. Implementation of Sensor Integration Model

Following the implementation of smart contracts on Solidity Remix and the deployment of these contracts on Ganache, the MetaMask extension was installed on the browser. MetaMask was then connected to Ganache, enabling all transactions to be processed through MetaMask-linked accounts from Ganache. In the Ganache test environment, some test accounts contains test Ethers for testing transactions. For users to interact with our proposed model, we have deployed the web application and APIs on our local PC, deployed the database on the SQL server in 2019, and connected our web application and APIs with the database. The web application serves as a portal for sensor data providers and consumers. It contains operations like registration and Login/authentication, sensor registration, sensor data push, Getting sensor details, Getting transaction details for sensor data providers, and operations like registration and Login/authentication, payment, and searching sensor data for sensor data consumers. The detailed Sensor Integration Model Work Flow is illustrated in Figure 8.

 doi:10.20944/preprints202406.0230.v1

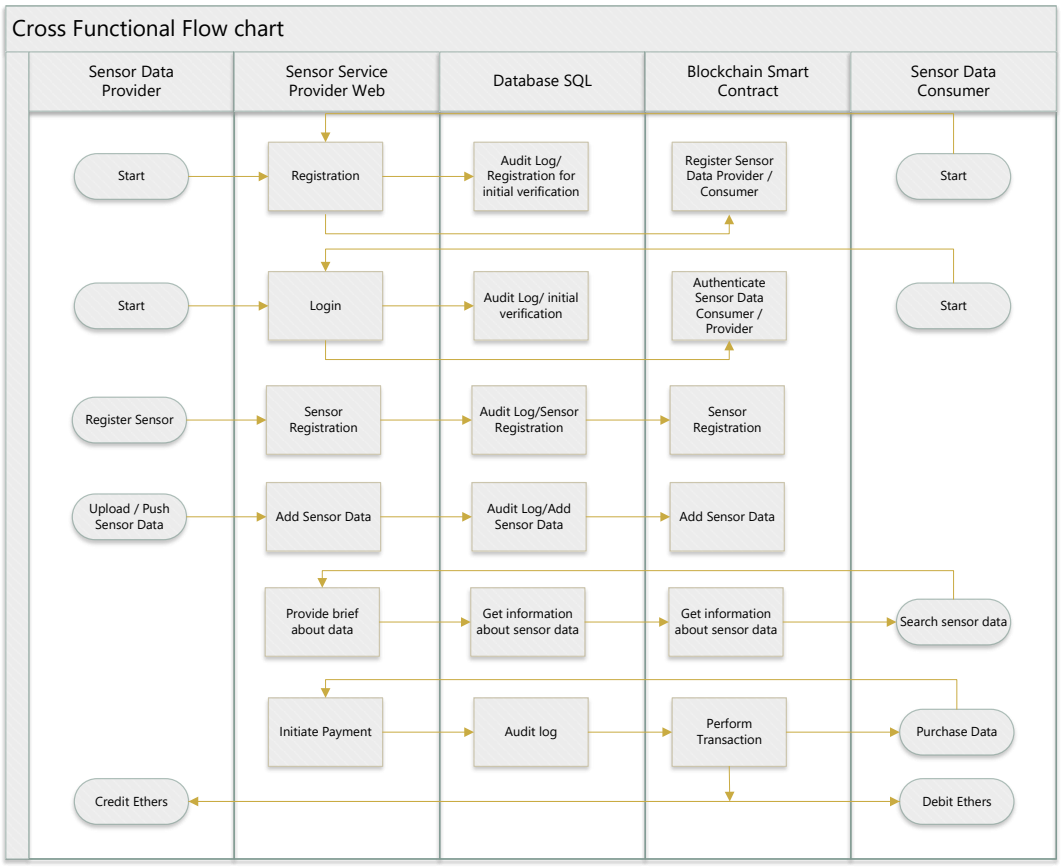

Figure 8. Sensor Integration Model Work Flow

4.3. Methods for Gathering Data

In our proposed model, there are two options for consuming the user's sensor data.

4.3.1. Sensor's Direct Push

In this method, we assume that the newly built sensors in the market can connect to our APIs, configure the schedule for pushing the data, register the sensors, register as a data provider through the sensors company portal or apps, and push the sensors data on APIs. For that purpose, we have developed rest APIs to perform the registration of data providers and sensors, upload and get the sensor's data, perform the transactions, etc

4.3.2. API Call

In this method, the user can extract their sensor data from their sensor management portal or data storage of the device, save the data in the file, and push the data file on our APIs.

4.4. APIs Methods and Types

Following the APIs that contain detailed information of its operations are proposed by our sensor integration model to integrate with any sensor for data provider registration and Login/authentication, sensor registration, sensor data push, Get sensor details, Get transaction details, data consumer registration and Login/authentication, payment, search sensor data. The APIs are developed on .Net core 6.0 with C sharp. All the APIs are rest APIs. ASP.Net core is a lightweight, high-performance, cost-effective approach that provides a modular HTTP request pipeline.

 doi:10.20944/preprints202406.0230.v1

4.5. Evaluation Metrics

This section examines the capabilities of the Sensors Integration Model and highlights its aspects based on the initial information collected. The following characteristics are discussed to validate our proposed framework.

- The computational and communication cost based on smart contract gas cost.
- Fairness of the proposed model.
- Security analysis between traditional sensors data providing

4.5.1. The Computational and Communication Cost Based on Smart Contract Gas Cost

The proposed solution is evaluated using the following performance metrics of communication and computational cost based on the gas cost consumption of smart contracts.

- The amount of gas cost consumed by smart contracts
- Transaction cost.
- Execution cost.

Each function's gas consumption is described in full, along with the cost in ethers and dollars. Equation 1 is used to determine the total cost in GWEI[1].

$$TotalCost(gwei) = GasUsed \times GasCost \tag{1}$$

For the purpose of the smart contract cost test, the standard cost of gas has been selected at 10 GWEI. Given that each function uses a different amount of gas, each function's cost is also different.

4.5.2. Fairness of the Proposed Model

One of the aspects of the evaluation matrix of our proposed model is fairness. Our proposed model achieves fairness in regular trade without the intervention of any third party. However, the sensor service provider must take action when the claim occurs. In the other situation, the security of the smart contract and our proposed system's invulnerability guarantee the sensor owner's fairness.

4.5.3. Security Analysis Between Traditional Sensors Data Providing

Our proposed sensor integration model also compares security analysis with traditional sensor data. We will evaluate our system with conventional systems based on identity management, traceability, confidentiality, blockchain storage, universality, and fairness.

5. Experimental Results and Performance Analysis

The evaluation of the Sensor Integration Model and the key conclusions of our research are presented in this section. The performance of the sensor integration model is first evaluated based on the submitted results. To evaluate the effectiveness of our model, we calculate the computational and communication costs of smart contracts based on gas costs. Second, we present a comparative security analysis of traditional sensors and data-providing systems. The section is divided into three sections. Section 5.1 provides the execution of the experimental setup. Section 5.2 presents a performance evaluation based on computational and communication costs. Section 5.3 elaborates on the fairness of the proposed model as compared to other systems. Section 5.4 presents the security analysis of traditional sensors and data-providing systems.

---

[1] Gwei (pronounced "gwee") is a unit of the cryptocurrency Ether (ETH), used on the Ethereum blockchain network.

 doi:10.20944/preprints202406.0230.v1

5.1. Experimental Setup Execution

As discussed, the sensor integration model deployment setup is in section 5. Now, we will experiment with it. We will experiment with the use case of a user who wears a smartwatch that contains sensors to get data on the user's heartbeats, sleeping time, the number of steps he walks in a day, blood pressure, etc. We have gathered the sensor dataset of the Fitbit device to perform our experiment. First, we register the user as a sensor data provider in the sensor service provider portal. Then, we register the Fitbit device sensor in the list of user sensors. We have experimented with uploading the data using both integration methods.

5.1.1. Sensor's Direct Push

In this method, after the sensor has been registered, we upload the sensor data on the portal using the user interface by providing its details. After the data is pushed successfully, it is ready to sell.

5.1.2. API Call

In this method, after the sensor has been registered, we upload the sensor data using the API on hourly basis for one day. After the data is pushed successfully, it is ready to sell. Then, we registered as sensor data consumers by providing all the details. We searched for the Fitbit sensor data. After the data appeared in the search results, we purchased the data. After the transaction was performed successfully, we received the data.

5.2. Computational and Communication Cost

In our prototype, bitcoin miners receive the fees for the gas needed to run our proposed sensor integration model system. Specifically, in our prototype, the gas price limit was set at $3,000,000 gas$. The rate of the ether at the time of the experiment is $1,901.87 USD$ per Ether. Figure 9, 10, 11 shows the result of gas cost, transaction cost and execution cost consumed by each function, respectively. We have seen that the largest gas cost is consumed by deploying and implementing smart contracts, which is about $2692355 gas$. The rest of the functions do not consume gas costs of more than $36040 gas$. In addition, since this smart contract deployment in our architecture is performed only once, the rest of the functions will be executed on every request of the sensor integration system.

Table 2. GAS cost consumed by smart contract

GAS Cost	
Deployment	2692355 gas
Sensor Data Provider Registration	296978 gas
Search Sensor Data Provider	0 gas
Register sensor	223260 gas
Search Sensor	0 gas
Sensor Data Consumer Registration	324451 gas
Get Sensor Data Consumer	0 gas
Send to contract	49820 gas
Balance Received to contract	0 gas
Contract to Address	36040 gas

 doi:10.20944/preprints202406.0230.v1

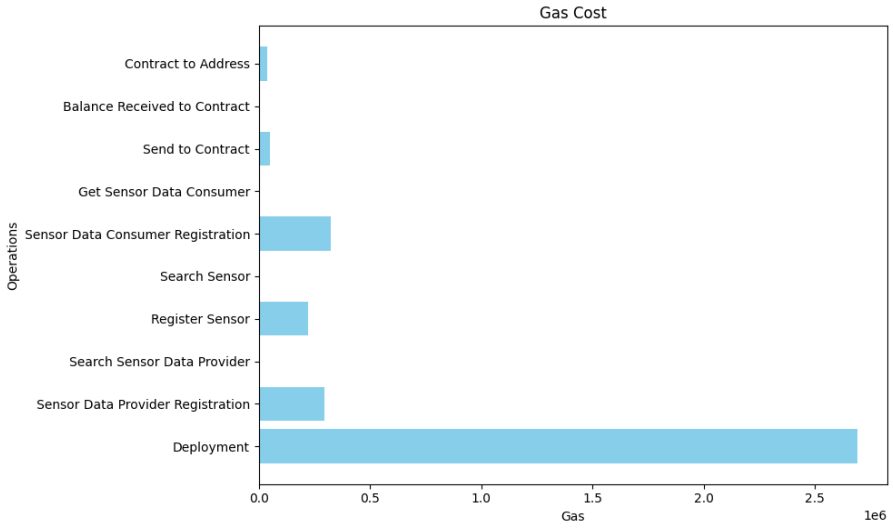

Figure 9. GAS Cost

Table 3. Transaction cost consumed by smart contract

Transaction Cost	
Deployment	2341842 gas
Sensor Data Provider Registration	258241 gas
Search Sensor Data Provider	0 gas
Register sensor	194139 gas
Search Sensor	0 gas
Sensor Data Consumer Registration	282131 gas
Get Sensor Data Consumer	0 gas
Send to contract	43321 gas
Balance Received to contract	0 gas
Contract to Address	31339 gas

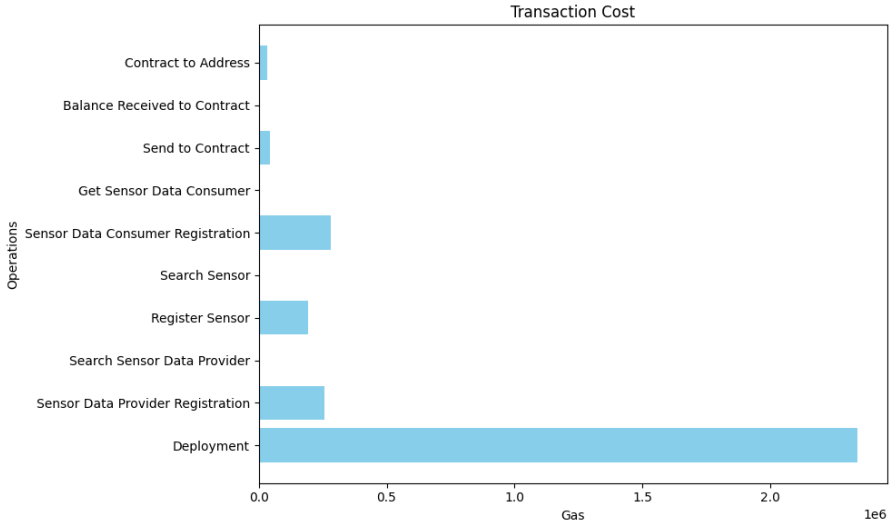

Figure 10. Transaction Cost

Table 4. Execution cost consumed by smart contract

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 June 2024 doi:10.20944/preprints202406.0230.v1

Execution Cost	
Deployment	2124602 gas
Sensor Data Provider Registration	234101 gas
Search Sensor Data Provider	21378 gas
Register sensor	170799 gas
Search Sensor	22272 gas
Sensor Data Consumer Registration	257535 gas
Get Sensor Data Consumer	257535 gas
Send to contract	22257 gas
Balance Received to contract	2496 gas
Contract to Address	9907 gas

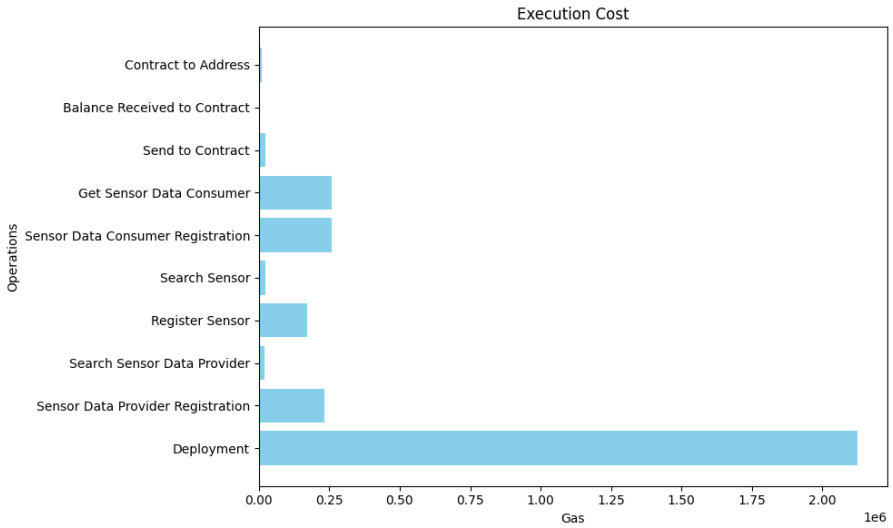

Figure 11. Execution Cost

5.3. Fairness of The Proposed Model

In regular trading, our proposed sensor integration model approach guarantees fairness without additional use by a third party. However, the sensor service provider must step in whenever the claim occurs. On the other side, the longevity of the sensor integration model system ensures the sensor owner's fairness and the smart contract's security and privacy. That is, without paying the necessary service price, no sensor data consumer can submit a request to get the data. As soon as the sensor data consumer understands that the received sensor data is useless, With the assistance of the sensor service provider, it can claim to have obtained the desired outcome or punish the sensor owner.

5.4. Security Analysis Between Traditional Sensors Data Providing Systems

In this section, we have compared our sensor integration model with some existing data storage and data sharing systems to elaborate on the comparison and benefits of our proposed system.
We have denoted the symbol below in Table 5.

Table 5. Comparison symbols

 doi:10.20944/preprints202406.0230.v1

Features	
Y	Supported
-	Non-Involving
N	Unsupported
Levels	
H	High
M	Medium
L	Low

The total amount of data saved in the blockchain is referred to as "Blockchain Storage." It contains three types: low, medium, and high. In this case, the level "H" is related directly to maintaining sensor data in the blockchain. "M" indicates storing ciphertext. L represents the linking of the hash data. Furthermore included in our definition of "universality" is the ability to accept continuing blockchain systems, such as Ethereum and Hyperledger, without requiring any changes. As shown by the comparison findings in Table 6, We can recognize that our proposed model provides both necessary security and confidentiality as well as decentralized fairness. This indicates that our sensor integration model can meet all the requirements required to store sensor data systems in smart cities and, hence, is superior to other alternatives that have been previously suggested.

Table 6. Comparison with other existing solutions

Property	Our proposal	Han-Yu et al. [20]	Al Sadawi et al. [25]	Yinghui et al. [18]	Jin-Ghoo et al. [23]	Shuang et al. [21]
Identity Management	Y	N	N	Y	N	Y
Pseudonymity	Y	N	N	Y	Y	Y
Confidentiality	Y	Y	Y	Y	Y	N
Traceability	Y	-	-	Y	N	N
Fairness	Y	N	N	N	N	N
Universality	Y	Y	N	N	N	N
Blockchain storage	M	M	L	M	H	L

6. Conclusions

In conclusion, this study makes a substantial contribution to the understanding of the challenges associated with handling Internet of Things (IoT) sensor data in smart city settings. This study attempts to overcome financial and technological barriers while putting data security, confidentiality, safety, and fairness first by introducing a comprehensive Sensor Integration Model for Secure Data Sharing and Storage. The suggested approach shows how to use smart contract systems and user-friendly interfaces, like an encrypted database and online application site, to simplify data management and maintain strict security guidelines. Furthermore, it is verified that the suggested model is more advanced than current alternatives and that it is technically feasible through extensive security research and comparative evaluations. This study highlights how important it is to involve stakeholders and encourage equity in data exchanges within the context of smart city ecosystems. Stakeholders' continued involvement and financial support of the suggested model can aid in its continuous improvement and success. In the future, research projects might concentrate on improving fairness mechanisms even more, reducing dependency on sensor service providers, and investigating cutting-edge encryption methods to improve database security and efficiency. In summary, this study not only provides a workable answer to the problems that smart cities are currently facing with sensor data integration, but it also opens the door for future advancement and innovation in this vital area.

Author Contributions: Conceptualization, A.R.S.K. and H.J.S.; methodology, H.J.S. and J.S.; software, J.S. and A.K. ; validation, H.J.S, J.S. and A.K.; formal analysis, H.J.S. and J.S.; investigation, A.R.S.K. and H.J.S.; resources, H.J.S. and J.S.; data curation, A.R.S.K. , H.J.S. and J.S.; writing—original draft preparation, A.R.S.K. and H.J.S.; writing—review and editing, F.Q. and Q.N.N.; visualization, X.X.; supervision, H.J.S. and F.Q.; project administration, F.Q.

 doi:10.20944/preprints202406.0230.v1

and Q.N.N.; funding acquisition, F.Q. and Q.N.N. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Universiti Kebangsaan Malaysia, Fundamental Research Grant Scheme having Grant number FRGS/1/2023/ICT07/UKM/02/1 and FRGS/1/2022/ICT11/UKM/02/1. The research was also supported by Posts and Telecommunications Institute of Technology Research Grant.

Institutional Review Board Statement: Not applicable

Informed Consent Statement: Not applicable

Data Availability Statement: Not applicable

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Hassan, R.; Qamar, F.; Hasan, M.K.; Aman, A.H.M.; Ahmed, A.S. Internet of Things and its applications: A comprehensive survey. *Symmetry* **2020**, *12*, 1674.
2. Haras, M.; Skotnicki, T. Thermoelectricity for IoT–A review. *Nano Energy* **2018**, *54*, 461–476.
3. Wei, Y.; Yuan, H.; Li, H. Exploring the Contribution of Advanced Systems in Smart City Development for the Regeneration of Urban Industrial Heritage. *Buildings* **2024**, *14*, 583.
4. Joyce, A.; Javidroozi, V. Smart city development: Data sharing vs. data protection legislations. *Cities* **2024**, *148*, 104859.
5. Almalki, F.A.; Alsamhi, S.H.; Sahal, R.; Hassan, J.; Hawbani, A.; Rajput, N.; Saif, A.; Morgan, J.; Breslin, J. Green IoT for eco-friendly and sustainable smart cities: future directions and opportunities. *Mobile Networks and Applications* **2023**, *28*, 178–202.
6. Ismagilova, E.; Hughes, L.; Rana, N.P.; Dwivedi, Y.K. Security, privacy and risks within smart cities: Literature review and development of a smart city interaction framework. *Information Systems Frontiers* **2022**, pp. 1–22.
7. Kamal, S.M.A.; Kafi, N.; Samad, F.; Syed, H.J.; Durrani, M.N. Modelling Civic Problem-Solving in Smart City Using Knowledge-Based Crowdsourcing. *International Journal of Computer Science & Network Security* **2023**, *23*, 146–158.
8. Wenhua, Z.; Qamar, F.; Abdali, T.A.N.; Hassan, R.; Jafri, S.T.A.; Nguyen, Q.N. Blockchain technology: security issues, healthcare applications, challenges and future trends. *Electronics* **2023**, *12*, 546.
9. Bahrepour, D.; Maleki, R. Benefit and limitation of using blockchain in smart cities to improve citizen services. *GeoJournal* **2024**, *89*, 57.
10. Chaganti, R.; Varadarajan, V.; Gorantla, V.S.; Gadekallu, T.R.; Ravi, V. Blockchain-based cloud-enabled security monitoring using internet of things in smart agriculture. *Future Internet* **2022**, *14*, 250.
11. Farooq, K.; Syed, H.J.; Alqahtani, S.O.; Nagmeldin, W.; Ibrahim, A.O.; Gani, A. Blockchain federated learning for in-home health monitoring. *Electronics* **2022**, *12*, 136.
12. Haque, A.B.; Bhushan, B.; Dhiman, G. Conceptualizing smart city applications: Requirements, architecture, security issues, and emerging trends. *Expert Systems* **2022**, *39*, e12753.
13. Kazmi, S.H.A.; Qamar, F.; Hassan, R.; Nisar, K. Improved QoS in Internet of Things (IoTs) through Short Messages Encryption Scheme for Wireless Sensor Communication. 2022 International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS). IEEE, 2022, pp. 1–6.
14. Hemamalini, V.; Mishra, A.K.; Tyagi, A.K.; Kakulapati, V. Artificial Intelligence–Blockchain-Enabled–Internet of Things-Based Cloud Applications for Next-Generation Society. *Automated Secure Computing for Next-Generation Systems* **2024**, pp. 65–82.
15. YR, S.K.; Champa, H. An extensive review on sensing as a service paradigm in iot: Architecture research challenges lessons learned and future directions. *Int. J. Appl. Eng. Res* **2019**, *14*, 1220–1243.
16. Alam, M.; Emmanuel, N.; Khan, T.; Xiang, Y.; Hassan, H. Garbled role-based access control in the cloud. *Journal of Ambient Intelligence and Humanized Computing* **2018**, *9*, 1153–1166.
17. Zhou, L.; Varadharajan, V.; Hitchens, M. Integrating trust with cryptographic role-based access control for secure cloud data storage. 2013 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications. IEEE, 2013, pp. 560–569.
18. Li, J.; Zhang, Y.; Chen, X.; Xiang, Y. Secure attribute-based data sharing for resource-limited users in cloud computing. *Computers & Security* **2018**, *72*, 1–12.

 doi:10.20944/preprints202406.0230.v1

19. Li, J.; Yao, W.; Zhang, Y.; Qian, H.; Han, J. Flexible and fine-grained attribute-based data storage in cloud computing. *IEEE Transactions on Services Computing* **2016**, *10*, 785–796.
20. Lin, H.Y.; Hung, Y.M. An improved proxy Re-encryption scheme for IoT-based data outsourcing services in clouds. *Sensors* **2020**, *21*, 67.
21. Sun, S.; Du, R.; Chen, S.; Li, W. Blockchain-based IoT access control system: towards security, lightweight, and cross-domain. *IEEE Access* **2021**, *9*, 36868–36878.
22. Bentahar, A.; Meraoumia, A.; Bradji, L.; Bendjenna, H. Sensing as a service in Internet of Things: efficient authentication and key agreement scheme. *Journal of King Saud University-Computer and Information Sciences* **2022**, *34*, 5493–5509.
23. Noshad, Z.; Khan, A.U.; Abbas, S.; Abubaker, Z.; Javaid, N.; Shafiq, M.; Choi, J.G. An Incentive and Reputation Mechanism Based on Blockchain for Crowd Sensing Network. *Journal of Sensors* **2021**, *2021*.
24. Dowlatshahi, M.B.; Rafsanjani, M.K.; Gupta, B.B. An energy aware grouping memetic algorithm to schedule the sensing activity in WSNs-based IoT for smart cities. *Applied Soft Computing* **2021**, *108*, 107473.
25. Al Sadawi, A.; Hassan, M.S.; Ndiaye, M. A survey on the integration of blockchain with IoT to enhance performance and eliminate challenges. *IEEE Access* **2021**, *9*, 54478–54497.
26. Liu, Q.; Zhang, F.; Qu, L. Application Research of Intelligent Site Management System. 2023 Smart City Challenges & Outcomes for Urban Transformation (SCOUT). IEEE, 2023, pp. 201–205.
27. Alam, T. Cloud-based IoT applications and their roles in smart cities. *Smart Cities* **2021**, *4*, 1196–1219.
28. Pradhan, N.R.; Singh, A.P. Smart contracts for automated control system in blockchain based smart cities. *Journal of Ambient Intelligence and Smart Environments* **2021**, *13*, 253–267.
29. Sampaio, S.; Sousa, P.R.; Martins, C.; Ferreira, A.; Antunes, L.; Cruz-Correia, R. Collecting, processing and secondary using personal and (pseudo) anonymized data in smart cities. *Applied Sciences* **2023**, *13*, 3830.
30. Xie, Q.; Li, K.; Tan, X.; Han, L.; Tang, W.; Hu, B. A secure and privacy-preserving authentication protocol for wireless sensor networks in smart city. *EURASIP Journal on Wireless Communications and Networking* **2021**, *2021*, 119.