Journal of Namibian Studies, 35 (2023): 222-253 ISSN: 2197-5523 (online)

Special Issue on Engineering, Technology and Sciences

Securing The Future Of Smart Farming: How Blockchain And IPFS Can Protect Sensor Data For Sustainable Precision Agriculture

Lamaa Sellami[1], Mounira Tarhouni[2], Bechir Alaya*[3], and Pascal Lorenz[4]

[1]Numerical Control of Industrial Processes Laboratory (CONPRIS), National School of Engineers of Gabes, Gabes University, Tunisia
[2]Higher Institute of Computer Science and Multimedia Gabes, Gabes University, Tunisia
[3]Department of Management Information Systems and Production Management, College of Business and Economics, Qassim University, 6633, Buraidah, 51452, Saudi Arabia.
[4]Université de Haute-Alsace (UHA), MIPS, France

Abstract

Blockchain technology has shown promise for applications beyond cryptocurrencies by enabling new forms of decentralized data management and value exchange. One sector that could benefit is agriculture through innovations in precision monitoring and resource allocation. Traditionally, smart farming systems rely on centralized platforms to integrate sensor data from fields and facilities. However, this model presents challenges around data security, access control and single points of failure. By developing a prototype combining blockchain and Internet of Things (IoT) technologies, we aimed to address these limitations and validate an alternative approach. Our solution integrated moisture sensors with a wireless network module to securely transmit real-time measurements to a distributed ledger. This allowed recorded data to be accessed remotely via a mobile application. By demonstrating this decentralized protocol for collecting, recording and viewing sensor data, our work indicates the potential of blended blockchain and IoT technologies to enable transparent and long-term sharing of

agricultural insights. Moving away from centralized systems through this proof-of-concept paves the way for more resilient and sustainable models of precision agriculture informed by distributed sensor records over time. In summary, our prototype provided a working example of how integrating blockchain into an IoT network can validate new governance frameworks for agricultural data storage and access beyond traditional cloud-dependent models. This offers opportunities to further optimize farming resources and operations through transparent sensor-driven decision making.

Keywords: blockchain; Internet of Things; smart farming; security; MQTT; IPFS

1. Introduction

The Internet of Things (IoT) is one of the fastest growing technology industries currently, with applications spanning multiple domains including automotive, smart home and healthcare [1]. IoT networks are often characterized by billions of connected devices, each producing massive amounts of data. However, there are challenges to ensure the scalability, security and privacy of this data as networks expand exponentially [2]. One approach is to use a decentralized file storage and distribution system called an InterPlanetary File System (IPFS) [3]. An IPFS allows data to be stored and shared across a network of nodes without a central point of failure or control. It addresses the scalability and performance issues of centralized storage models by utilizing a content-based hash addressing system and incentive-based distribution of data across nodes. Records are identified by their hash rather than location, enabling high throughput, fault tolerance and wide distribution of petabytes of data.

Blockchain technology is also decentralized in nature and provides security and privacy guarantees for transactions through mechanisms like digital signatures and smart contracts executed across distributed ledgers [4]. Applications of blockchain for IoT include decentralized identity management to authenticate devices and users, access management to control data sharing permissions and secure transaction processing to enable micropayments between devices and businesses involved in data production and consumption [5]. Together, IPFS and blockchain offer promising solutions to

handle the massive growth of data in scalable, private and trustless ways within complex IoT ecosystems. By leveraging the strengths of these different technologies, an IoT–IPFS–blockchain combination has the potential to provide end-to-end solutions for IoT challenges [6]. It allows data to be stored and shared in a decentralized, secure manner using IPFS distributed file storage. Identities and access permissions can be managed transparently through blockchain-based mechanisms. Smart contracts on the blockchain also ensure data integrity and authenticity. With IPFS providing the scalable data distribution and storage layer, and blockchain offering security, transparency and programmability, this hybrid approach could empower applications dealing with the massive volumes of data generated by IoT networks [7]. Issues around data provenance, privacy and access control could be addressed seamlessly. Furthermore, it establishes an open and interoperable framework for data sharing in a way that preserves user ownership and privacy [8] over their own information.

This article explores the opportunities of combining IoT, IPFS, and blockchain technologies to address the challenges of IoT, with a focus on an automatic irrigation system. It is organized into three parts. The first part introduces the architecture of an IoT system and discusses the security issues associated with this technology. The second part explains the advantages and disadvantages of an IPFS and blockchain technologies. IPFS enables decentralized and secure storage and sharing of data, while blockchain provides a transparent and immutable ledger for secure transactions and identity management through smart contracts. The third part focuses on applying the IoT–IPFS–blockchain combination to an automatic irrigation system. It details the proposed architecture, which includes IoT sensors for collecting environmental data, an IPFS network for decentralized storage and data sharing and a blockchain for ensuring transaction security and data integrity. The benefits of using an IPFS for irrigation data storage are discussed, including data availability despite node failures and integrity through content-based addressing. The advantages of using blockchain, such as data traceability and secure process execution through smart contracts, are also examined. The potential drawbacks of using these technologies are also discussed, such as IPFS performance and latency issues, as well as the computational and storage requirements of blockchain. Finally, the implementation of the solution is presented, describing the configuration of an IoT system, the integration of

IPFS for decentralized storage and the deployment of a blockchain network with smart contracts for secure data and transaction management. This combination of technologies provides robust data protection, ensures the integrity and security of the automatic irrigation system and enables reliable and trusted operations. In the context of automatic irrigation systems, integrating blockchain and an IPFS addresses several existing challenges. Prior to leveraging these technologies, these systems face issues such as lack of transparency and trust in data, vulnerability to sensor data tampering and difficulties in ensuring data integrity and traceability. By utilizing blockchain, an immutable and transparent ledger can be created, ensuring data integrity and accountability. The integration of IPFS provides decentralized and reliable storage for sensor data, enabling easy access and retrieval of historical data. Together, these technologies enhance water usage, irrigation planning and data-driven decision-making, promoting sustainable agricultural practices and optimized crop management in automatic irrigation systems.

In conclusion, the article highlights the advantages and challenges of using IoT, IPFS and blockchain in the context of an automatic irrigation system and proposes an integrated approach to address these challenges and enhance the security and efficiency of the system.

2. Related Work

The Internet of Things (IoT) emerged due to advancements in technologies that enabled connectivity between devices. The IoT involves integrating digital capabilities into physical objects through sensors, software and network connectivity. This allows objects to transmit data autonomously. The IoT has grown substantially, with projections of over 20 billion active connected devices globally by 2025. Bringing internet access to physical objects through embedded sensors and networks promises efficiency and operational improvements across many domains. In short, the IoT has flourished by enabling digital interactions between devices in the real world.

Several studies have investigated the use of the IoT and smart sensor technologies in the field of smart agriculture [9,10]. These studies have highlighted the advantages of deploying a network of sensors and devices across farmland to gather real-time data on various environmental factors. Parameters such as soil moisture, temperature, humidity, light levels and nutrient content can be continuously monitored,

providing farmers with valuable insights into the health and growth of their crops.

In [11], the authors demonstrate that by leveraging the IoT, farmers can make informed decisions regarding irrigation, fertilization and pest control. For example, IoT-enabled systems can automatically trigger irrigation when the soil moisture level falls below a specific threshold, ensuring that crops receive sufficient water [11]. This data-driven approach allows farmers to optimize their farming practices and enhance crop yields. In research conducted by Badran and Kim [12,13], IoT networks were proposed to monitor environmental factors and soil moisture, providing farmers with information to make effective decisions about irrigation and crop yield. Kim's study specifically aimed to address challenges faced by Indian farmers through the development of a low-cost smart farming solution. On the other hand, Ramli and Lova [14,15] explored the potential benefits of smart soil and air sensors, along with sophisticated IoT applications, to provide farmers with tailored advice and accurate sensor readings based on an analysis of 60 research articles. The overall objective of these studies was to develop affordable and effective connected farming solutions by integrating diverse data sources and sensors to optimize production and support farmers, although some systems may have higher energy usage or costs associated with them.

As represented by [16], in the agriculture industry, IoT devices are used to monitor the well-being of animals by tracking activity levels, heart rate, body temperature and behavior patterns. The data are transmitted to a central system and analyzed in real-time, allowing farmers to access the information and detect any signs of distress or illness. This enables timely intervention and treatment, reducing the spread of diseases within the herd. IoT technology also plays a role in supply chain management by tracking the conditions of harvested produce during storage and transportation [17]. Devices measure factors such as temperature, humidity and air quality, ensuring optimal storage conditions and minimizing the risk of spoilage or waste.

In addition, wireless sensor networks have been explored for environmental monitoring and disaster response. This includes underwater wireless communication [18], pipeline monitoring [19] and tracking air pollution [20]. Wireless sensor technology has also been proposed for natural disaster monitoring [21], with optimizations for energy efficiency, low costs, network protocols and intelligent communications to

support geohazard analysis and risk mitigation.

Overall, the research suggests that wireless sensor technology has the potential for diverse environmental and infrastructure surveillance applications across sectors such as agriculture, pollution tracking [22] and emergency response, requiring real-time data collection and analysis from both underwater and terrestrial deployments [23].

In [24], the authors gave a specific view on IoT applications which encompasses physical devices, gateways, edge nodes, cloud infrastructure and the internet. However, in order to fully exploit the potential of IoT, standardized protocols and architecture layers are being developed to provide comprehensive services to IoT system devices. Various weaknesses inherent in centralized IoT architectures have been identified by [25,26]. Currently, the dominant approach to IoT architecture is centered on a centralized model, where data are collected from edge devices and transmitted to a central server or cloud for processing. However, this centralized paradigm faces several limitations and vulnerabilities that require special attention [27,28].

The current centralized nature of IoT infrastructures presents challenges in terms of reliability, privacy and latency [29,30]. However, there is a growing trend towards decentralized systems leveraging blockchain technology to address these issues [31]. By incorporating blockchain, IoT applications can benefit from enhanced security, improved privacy, increased trust and reduced latency [32,33].

The use of blockchain in IoT enhances security by leveraging its immutable and verifiable nature, ensuring the integrity of data. The decentralized and transparent approach provided by blockchain eliminates the need for a central authority, reducing the risk of unauthorized access or data breaches. Blockchain also enables the secure and tamper-resistant storage of IoT data, treating messages exchanged between smart objects as transactions. This strengthens the defense against cyberattacks [34] and unauthorized tampering. Furthermore, blockchain serves as a foundational layer for IoT applications involving transactions and interactions, with smart contracts playing a crucial role in this integration [35–37]. Smart contracts are gaining popularity in various IoT applications [38], enabling the management of Quality of Service (QoS) [39] by ensuring reliable and accurate data for measuring the quality of IoT services [40–42] (see Table 1). Overall, the adoption of

blockchain technology has the potential to bring significant transformation to the IoT industry, addressing concerns related to security, privacy, trust and latency [43]. By embracing decentralized and distributed infrastructures, IoT applications can unlock the advantages offered by blockchain and pave the way for more robust and efficient IoT systems [44].

Table 1. Comparison between IoT and blockchain

Characteristic	IoT	Blockchain
Structure	Centralized (Cloud IoT model)	Decentralized
Reliability	Single point of failure (by example, DoS attack on cloud servers)	More reliable because it eliminates single points of failure
Resources	Limited resources with a number of limited bandwidth and calculations	High resource consuming and bandwidth consumption
Scalability	Integrates billions of devices	Suffers from low throughput and low transaction speed
Automation	Requires human intervention	Autonomous interactions governed by smart contracts
Latency	Requires low latency	Mining takes time time
Traceability	Low traceability	Inviolable
Transparency	Low transparency	Trace any past transactions
Interoperability	Heterogeneous system with poor interoperability	Improves interoperability
Data sharing	Data exchange is a difficult task	Data sharing can be achieved via a blockchain layer P2P with uniform access across different IoT systems
Security	Vulnerable security and malicious attacks	All transactions are encrypted and digitally signed by cryptographic keys [45]. Blockchain can be used to automatically update IoT firmware

3. Proposed Approach

3.1. Methodology

This study aims to explore how integrating IoT, IPFS and blockchain technologies can address security and data management challenges for an automatic irrigation IoT system.

The methodology involves designing an architecture that leverages the security features of IPFS decentralized data storage and blockchain distributed ledger technology. Environmental sensor data collected by the IoT devices will be stored on an IPFS in a decentralized manner to ensure availability. Blockchain will provide an immutable record of irrigation transactions for traceability and integrity.

To implement the proposed architecture, the following key

steps will be taken:

1. Develop the IoT infrastructure with sensors, actuators and a network gateway.
2. Integrate IPFS to distribute and store sensor data across nodes in the P2P network using cryptographic hashes.
3. Deploy a blockchain network with nodes validating and recording transactions in a distributed ledger.
4. Develop smart contracts to define automation rules and conditions for executing transactions like irrigation requests.

The proposed combination of IoT, IPFS and blockchain aims to address security and data management challenges through features such as decentralized storage, data integrity and transparent automation.

A blockchain-based implementation offers distinct advantages over a distributed database enforced with authorization mechanisms. Blockchain's decentralized nature ensures trust and transparency, as data are validated and stored across multiple nodes. Its immutability and tamper-resistant properties safeguard the integrity of data, while enhanced security measures provide protection against unauthorized access. The transparency and auditability of blockchain promote accountability, while disintermediation reduces costs. Additionally, blockchain ensures data consistency and synchronization among participants, making it an appealing choice for applications requiring trust, integrity and verifiability in a distributed setting.

The methodology involves designing, implementing and evaluating this integrated architecture for an automatic irrigation use case.

3.2. Enabling Technologies

In IoT systems, communication protocols are used to connect devices in the IoT. These protocols enable devices to communicate with each other and with other devices or services, such as servers and applications. Common examples of communication protocols used in IoT include HTTP, CoAP, AMQP and MQTT. Each protocol has specific features and capabilities for efficient and secure communication in IoT applications [46].

HTTP is widely used for web-based communication and can be employed in IoT applications for data exchange. CoAP is a lightweight protocol designed for devices with limited

resources. It facilitates communication between IoT devices using simple request and response messages. AMQP provides reliable, secure and efficient message delivery, supporting advanced features like message scheduling and routing. MQTT, known for its lightweight nature, is widely used for real-time data transmission in low-bandwidth networks, making it popular in IoT applications.

The choice of protocol depends on factors such as the specific use case, device capabilities, network constraints and desired reliability and efficiency levels. MQTT, for instance, is often selected for its lightweight design, efficiency and suitability for low-bandwidth and resource-constrained environments.

Regarding security, MQTT incorporates mechanisms such as Transport Layer Security (TLS) for ensuring secure communications. TLS encrypts and authenticates data exchanged between MQTT clients and brokers (See Figure 1), preventing unauthorized access and eavesdropping. By implementing TLS, MQTT establishes a secure channel, ensuring the confidentiality and integrity of information.

Figure 1. MQTT Broker.

In summary, communication protocols commonly used in the IoT include HTTP, CoAP, AMQP and MQTT. Each protocol offers specific features and can be chosen based on the requirements of the IoT application in terms of bandwidth, resources, reliability and security. MQTT stands out for its lightweight nature, efficiency and ability to operate in low-bandwidth environments. It also incorporates security mechanisms like TLS for secure communications between IoT devices and message brokers.

As shown in Figure 2, in the proposed IoT system, the MC node and the soil sensor publish the data on a subscribe topic

using the Message Queuing Telemetry Transport (MQTT) protocol. We integrate a blockchain network and IPFS to secure the captured data against IoT attacks.

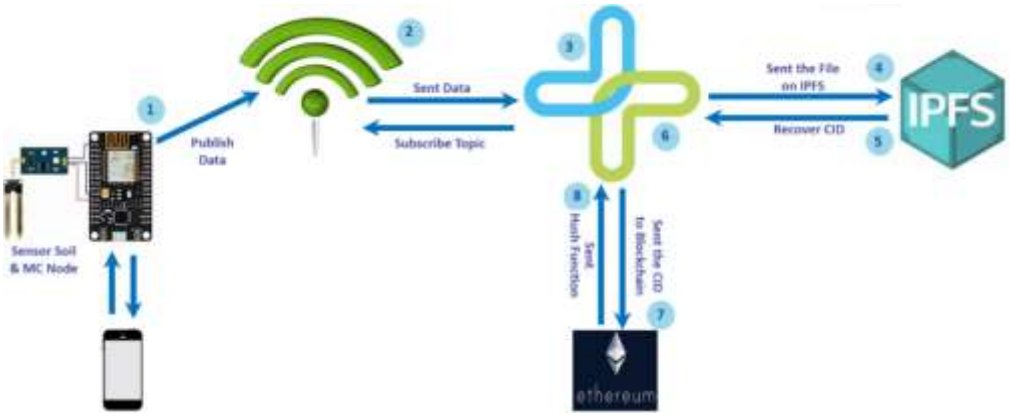

Figure 2. General structure of our proposed approach: It involves setting up an IoT system (irrigation system). This system sends data via the MQTT protocol. We integrated this system with a blockchain network and IPFS to secure the captured data against IoT attacks.

Here are the detailed steps of the system:

- The soil sensor collects data related to soil moisture.
- The MC node receives the data from the sensor and publishes it on an MQTT subscribe topic.
- The IoT system retrieves the data published by the MC node via MQTT.
- The data are then sent to IPFS, where it is stored securely in a decentralized manner.
- IPFS generates a unique Content Identifier (CID) for the stored data. The CID is a unique address based on the content of the data.
- The CID is then sent to the blockchain, where it is recorded in a block.
- The blockchain returns a hashed function from the CID back to the IoT system.
- The IoT system receives the hashed function, which can be used to verify the integrity of the data stored on IPFS at any time.

The integration of the blockchain network and IPFS in the IoT system offers several advantages in terms of data security. Using IPFS ensures the decentralized and resilient storage of data, guaranteeing availability and recovery in case of a node failure. Additionally, recording the CID in the blockchain ensures data integrity, as any subsequent modification to the

data will result in a change in the corresponding CID.

In this IoT system, data collected by the soil sensor are published on an MQTT topic and then sent to IPFS for secure storage. The CID generated by IPFS is recorded in the blockchain to ensure data integrity. This integration between MQTT, IPFS and the blockchain provides a robust and secure solution for data collection, storage and management in the IoT system. We chose Ethereum as a reference blockchain technology which brings several advantages to the implementation. Ethereum is a widely recognized and established blockchain platform that supports smart contracts, enabling the execution of decentralized applications (DApps) with self-executing agreements. Its robust infrastructure and large developer community provide a wealth of resources and tools for building and deploying blockchain-based solutions. Ethereum's native cryptocurrency, Ether (ETH), facilitates secure and efficient transactions within the network. Additionally, Ethereum's open-source nature allows for customization and interoperability, making it a flexible choice for various use cases. By leveraging Ethereum, the implementation benefits from a mature ecosystem, extensive developer support and the ability to tap into the broader blockchain community.

3.3. Combining IPFS and Blockchain for IoT Security

The proposed technical architecture combines IPFS and blockchain at the level of gateways connected to IoT devices. The data generated by the devices are first stored in a decentralized manner on an IPFS through the gateways. Each IPFS block receives a unique hashed identifier that is then recorded in a transaction on the blockchain by the gateway. This allows each IoT device to be assigned a unique public address during initial configuration on the blockchain via its gateway. This address serves for authentication and identification of the device [47]. A digital signature algorithm like ECDSA is used to guarantee the integrity of exchanged data. Key metadata for each device (type, owner, location, etc.) is also redundantly stored on multiple IPFS nodes via the gateways, improving availability of information. The complete history of data generated and transactions performed by each IoT device can be traced end-to-end immutably on the blockchain. This ensures traceability and auditability of processes. An access control mechanism based on public addresses allows limiting and securing access to data according to the rights granted. In terms of scalability, some optimizations are possible such as grouping transactions by gateways or indexing data on an IPFS rather than full storage. Despite these measures, the limited capacities and autonomy of IoT devices remain a challenge for very large-scale deployment of this architecture. The redundant distribution of metadata and

immutable recording of activity on the blockchain increase the security, resilience and auditability of IoT systems compared to traditional centralized approaches.

IPFS is a decentralized and distributed file system designed to address the challenges of exponentially growing data on the internet. Unlike traditional file systems which centralize storage on servers, IPFS is based on a peer-to-peer architecture where files are distributed across the nodes of the network. When a file is added to IPFS, it is assigned a CID which is calculated from the file's cryptographic hash. This allows the system to ensure file integrity in a decentralized way, without relying on a single source of truth. Nodes communicate to share metadata about the physical location of files, which is stored in a decentralized registry (DHT). When a request for a file is made, nodes collaborate to locate those physically possessing it and organize transmission from peer to peer until it reaches the requester. This process is similar to torrent-style filesharing. Due to its distributed nature, IPFS provides many advantages such as redundant, resilient data storage and permanent access to content via unique CIDs, independent of central providers. Practical applications of IPFS include hosting decentralized websites, sharing multimedia content or exchanging data between Internet of Things devices in a resilient manner. The use of cryptographic identifiers and distributed data lookup allows files to be permanently stored and retrieved in a decentralized network without a single point of failure. Figure 3 explains how to send/obtain data in an IPFS system.

Decentralization is one of the key advantages of IPFS. As a decentralized storage network, IPFS is less vulnerable to attacks and single points of failure. It also distributes the workload and storage requirements across multiple computers. Another benefit is speed; IPFS uses a "chunking" technique to split files into smaller pieces, allowing faster downloading and uploading. This chunking also provides bandwidth efficiency by making it easier to locate and retrieve file pieces from various nodes. IPFS is designed to work well even in environments with lower internet speeds, providing resilient connectivity.

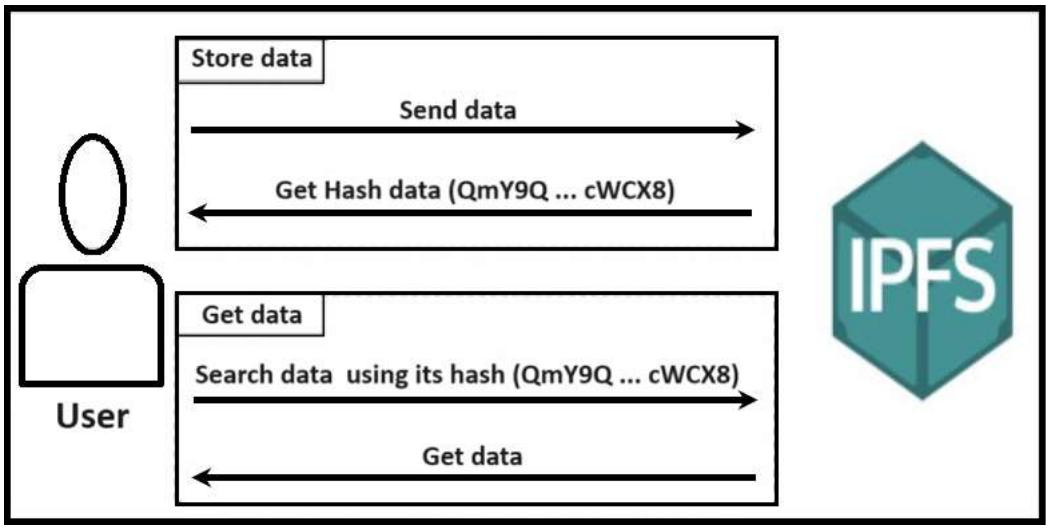

Figure 3. IPFS Functioning.

While IPFS offers many advantages over traditional centralized storage, it also faces some challenges. The setup and implementation of an IPFS network can be technically complex, especially for non-technical users unfamiliar with blockchain or distributed systems. As the technology is still in development with limited adoption, reliability and stability issues may occur. The decentralized nature of data storage across nodes also means users have less control over who accesses their files and how the data are used once shared on an IPFS. Immutability is not guaranteed as files can potentially be modified or deleted by altering the nodes storing those files. Widespread deployment of an IPFS will require addressing these disadvantages around control, reliability and ensuring permanence of stored content. An IPFS provides a decentralized solution to large-scale data storage and access but bringing the full benefits of the technology to mainstream use will involve ongoing development efforts to improve ease of use, robustness and security of the network**.**

Immutability is not guaranteed natively with IPFS alone, since it is possible to modify or delete objects by acting on the nodes. This is an important limitation that must be overcome for certain uses requiring the permanence and integrity of data over time. The idea of combining IPFS with a blockchain is very relevant in this regard. By recording the CIDs of IPFS objects on the blockchain in a decentralized and immutable manner, we provide several benefits:

- Guarantee of uniqueness and durability of CID identifiers thanks to the distributed register that is the blockchain.
- Permanent traceability and auditability of IPFS objects via their history on the blockchain.
- Better resistance to unauthorized modifications or deletions since proof of the existence and initial content of objects is kept securely on the blockchain.

- Possibility of deriving services around the management, monetization or certification of IPFS objects from their recording in the blockchain.

3.4. Blockchain Application

A blockchain is a decentralized digital ledger that securely records transactions in a permanent way. It consists of a series of blocks linked together in a secure manner. Each block contains a recent set of transactions that have been validated by the peer-to-peer network. To securely link each block to the previous one, a unique cryptographic hash is calculated for each block including the hash of the previous block in its calculation. This chaining of blocks through hashing makes it impossible to retroactively modify their contents without the network noticing. Indeed, altering the content of a block would break the coherence of the chain with subsequent blocks. Most blockchains function as state transition machines. Peers submit transactions that are verified and grouped into blocks by miners. Once validated, the new block is securely added to the blockchain. A freshly computed global state integrating the cumulative effects of transactions updates the previous one. Each node then synchronously updates its own copy of the blockchain state. This process reliably and permanently records, audits and secures any type of transaction between entities without a central authority. The integrity of the data is cryptographically secured through blockchain architecture (see Figure 4).

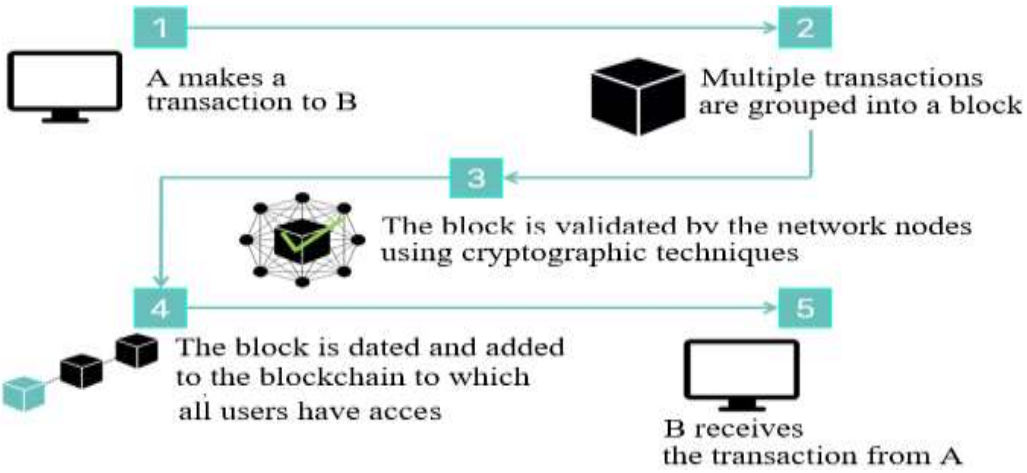

Figure 4. How blockchains work.

4. New Irrigation System

Automatic irrigation systems allow the remote control of watering for gardens, parks and plantations. They can be programmed to water on predefined schedules or based on

environmental parameters like soil moisture or temperature. These systems provide many benefits such as water savings by avoiding unnecessary watering, reduced workload from manual watering and irrigation tailored to plants' actual needs. For this work, we developed our own automatic irrigation system. It includes soil moisture sensors to monitor hydration levels, LEDs to simulate activating a water valve and an ESP8266 microcontroller to autonomously manage the system. The embedded microcode was developed using the MicroPython programming environment with the Upycraft IDE. Other hardware platforms like Arduino or other programming languages could also be suitable for this type of project. Our system demonstrates the concept of sensor-based irrigation automation to optimize water use and simplify garden maintenance. Precise watering controls can help conserve water resources for a sustainable future.

4.1. ESP8266 Card Configuration

To begin setting up the automatic irrigation system, we first connected the soil moisture sensor and an LED to the ESP8266 microcontroller card as illustrated in Figure 5. Next, we downloaded and installed the Upycraft programming software to allow flashing the MicroPython firmware onto the ESP8266. Once this was complete, we plugged the card into my computer via a USB. Following the initial hardware and software configurations, we were able to start writing Python code to read values from the moisture sensor.

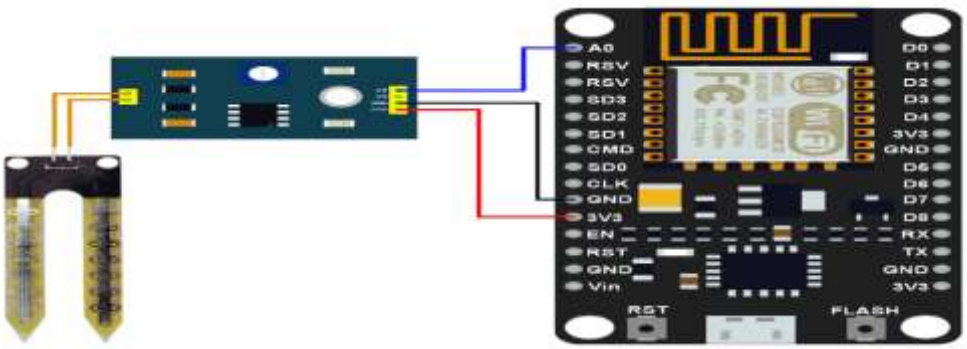

Figure 5. ESP8266 wiring with sensor of soil humidity

The code takes continuous readings and triggers the LED when soil moisture drops below a predefined threshold. This LED activation simulates the irrigation system turning on. Running the code confirmed it was functioning as intended, with the LED responding appropriately based on sensor values. This proved the ESP8266 platform could viably automate irrigation based on real-time moisture monitoring. The low-cost microcontroller and programming approach provides a starting point to remotely control watering based on sensor inputs.

Further testing of different threshold parameters and additional sensors should optimize the system for diverse plant needs and environments. While still undergoing refinement, this initial proof-of-concept validates using IoT technologies for sensor-driven irrigation to efficiently use water resources based on actual soil conditions.

We utilized the Adafruit IO platform to test the functionality of the MQTT communication protocol. Adafruit IO is an IoT development and testing platform that allows users to connect, control and monitor devices and sensors online.

Some key benefits of Adafruit IO include:

- it supports MQTT for bidirectional communication between devices and the cloud. This allows real-time data transfer from sensors to the broker.
- Users can easily visualize sensor data through configurable dashboards and graphs updated in real-time.
- Actuators can be remotely controlled by publishing messages to feed topics.
- The platform handles all the aspects of connecting to a broker and exchanging messages via MQTT topics.
- Project setup is straightforward without complex server configuration required.
- By linking the ESP8266 irrigation system to Adafruit IO, we were able to validate that soil moisture readings published via MQTT were being received properly in the cloud. Commands could also be sent to trigger the LED, demonstrating the end-to-end IoT connectivity powered by the MQTT protocol. Adafruit IO provided a simple test bed for developing and confirming our proof-of-concept automated irrigation system based on sensor data communication.

Figure 6 illustrates our simulation results in the Adafruit interface. We were able to visualize the soil moisture readings in real-time as waveforms updating automatically every time a new value was published by the ESP8266 system via MQTT. The dashboard also confirmed the LED was successfully triggering on and off based on the moisture threshold through the “State” feed. This validated the end-to-end functionality of gathering sensor measurements on the device, transmitting the data over WiFi via MQTT and receiving/parsing the messages in the cloud dashboard. Adafruit IO proved instrumental in demonstrating proper operation of the key IoT communication and sensor monitoring aspects of our automated irrigation prototype in a user-friendly interface.

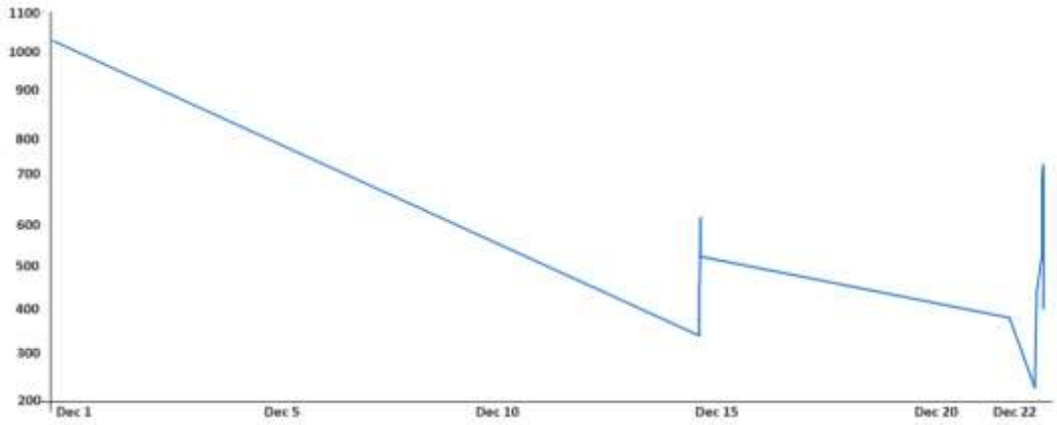

Figure 6. Dashboard Adafruit

4.2. Integrating IPFS in our IoT System

In our automatic irrigation system, precise soil moisture measurements are crucial for effectively monitoring plant hydration levels over time. To securely collect and archive these important sensor readings, we integrated IPFS technology into the design. IPFS is a peer-to-peer distributed file system that utilizes content-addressing and dispersal of hashes across nodes for robust data integrity. Our ESP8266 microcontroller runs a code to periodically sample the soil moisture sensor value. It then generates a hashed IPFS Object with the measurement data and metadata like the timestamp.

The microcontroller joins the IPFS network and pins this new Object to its local node. Pinning distributes copies of the Object across other connected nodes in the decentralized network. This provides fault tolerance, as even if the microcontroller node fails, the measurement will remain available from other peers. To retrieve data, we can connect to any participating IPFS node and request Objects using their hashes. As long as at least one node remains pinned, the immutable measurement record is retrievable in the future. Compared to centralized databases or cloud storage, IPFS offers higher assurances of long-term data availability and access without single points of control or failure. The hash-based addressing and cryptographic signatures inherent in IPFS also guarantee the authenticity and integrity of archived sensor logs. This confirms the moisture readings were not tampered with, giving users confidence in the historical accuracy of the dataset for analyzing irrigation needs over extended deployments. By leveraging IPFS, our system demonstrates how IoT applications can reliably log critical environmental readings in an open, distributed and tamper-proof manner at the network layer for superior data resilience.

When interacting with the IPFS network, one must first select an interaction method as there are several options available, each with their own pros and cons.

One of the most common approaches is using the IPFS Desktop client. It provides a user-friendly graphical interface for easily managing files and shared content on the network. However, it requires installation on a computer.

Alternatively, one can interact through the command line interface via a terminal emulator. While more technically complex than the desktop client, this method grants access to advanced capabilities and automation potential. It may prove difficult for novice users though. IPFS gateways serve as intermediaries between traditional web protocols and the decentralized IPFS network. They allow users to browse hashes and retrieve pinned files stored on IPFS through a standard web browser merely by navigating to a specific URL. For our project, we opted to leverage available IPFS gateways to access network content without needing a local node setup. This streamlined accessing moisture readings pinned to distributed hash table locations for long-term archiving. Gateway browsability validated proper remote data pinning and retrieval using only a browser from different points on the Internet. Thoughtful selection from interaction options like the desktop, command line or gateways is necessary when first engaging with an IPFS depending on one's technical skill level and task requirements. Each provides exclusive utilities but also constraints to consider.

Several researchers have studied how to integrate IoT and IPFS technologies to enhance privacy protection. In this phase, we will present our integration method which requires the use of an application acting as a mediator between the MQTT broker and IPFS network. To publish sensor data to an IPFS while preserving privacy, our method uses a mediator application deployed on edge infrastructure. The IoT devices (e.g., sensors) communicate privately with this local edge app using MQTT. The app collects and processes messages from devices, anonymizing any sensitive payload data before generating signed IPFS objects. It then pins these objects containing the anonymized data to its local IPFS node.

By pinning over the decentralized storage network, data availability and integrity are ensured even if the edge app is temporarily unavailable. However, as data transmission only occurs between devices and the local app, external parties have no visibility into raw sensor streams for node identification or profiling purposes. Compared to alternatives of exposing raw MQTT topics or uploading unprocessed payloads, our proposed integration architecture provides enhanced privacy safeguards by introducing the mediating application layer between IoT and distributed storage mechanisms. This approach could benefit privacy-critical deployments involving medical, transportation or smart home data.

We chose to use Python for developing our application. First, we installed the necessary packages using Flask to create our web application, the IPFS client for decentralized file

management, and Flask-MQTT to communicate with an MQTT broker. Flask is a micro web framework well-suited for building APIs and lightweight backend services. Its modular architecture helped accelerate development of the app mediation logic. The IPFS Python client enabled programmatically pinning and retrieving content from the distributed file system.

The Flask-MQTT extension provided a Pythonic interface to integrate MQTT communication with Flask endpoints. This allowed devices to securely send telemetry to the app before processing and persisting anonymized data to IPFS. Next, we connected our application to an MQTT broker using the Flask-MQTT package. We defined the connection parameters such as the server address, username and password. The Flask-MQTT client then handled the network communication with the broker using the MQTT protocol (see Figure 7). Once connected, we subscribed to topics where IoT devices publish readings. This enabled the application to receive new values in real-time.
We also published test messages to validate end-to-end flow from devices, through the broker and into our mediator layer for processing and IPFS storage.

```
app = Flask(__name__)
app.config['MQTT_BROKER_URL'] = 'io.adafruit.com' # your broker address goes here
app.config['MQTT_BROKER_PORT'] = 1883 # default port for non-tls connection
app.config['MQTT_USERNAME'] = 'raniabennasser' # No username set for now
app.config['MQTT_PASSWORD'] = 'aio_enia67QmlTiuU7WT2k3nl917zEFN' # no password set for now
app.config['MQTT_KEEPALIVE'] = 60 # keepalive every 5 seconds
app.config['MQTT_TLS_ENABLED'] = False # set TLS to disabled for testing purposes
mqtt = Mqtt()
mqtt.init_app(app)
running = True

@mqtt.on_connect()
def handle_connect(client, userdata, flags, rc):
    print("mqtt brocker connected ")
    mqtt.subscribe('raniabennasser/feeds/test.humidity') #your MQTT topic here
while running == True:
    @mqtt.on_message()
    def handle_mqtt_message(client, userdata, message):
        mqtt_data = message.payload.decode()
        print(f"Received MQTT data: {mqtt_data}")
```

Figure 7. MQTT server connection

Python proved practical for building this privacy-focused IoT/IPFS integration prototype. Its libraries supported common patterns like web frameworks, distributed storage and message brokering.. The simulation result is illustrated in Figure 8.

```
PS C:\Users\rania> & C:/Users/rania/AppData/Local/Programs/Python/Python311/python.exe c:/memoire.dossier/integrationPy.py
mqtt brocker connected
Received MQTT data: 420
```

Figure 8. MQTT results.

When sending moisture values to IPFS, our application first established a connection to the local IPFS node. This utilized the "Client" object provided by the ipfshttpclient package in Python. Upon receiving new sensor readings via MQTT, the application leveraged methods exposed by the IPFS client to archive the anonymized payloads in a distributed manner. Specifically, the add method was used to insert each payload as a new IPFS object stored within the node. The add method generated a CID hash representing the new file. These hashes were recorded along with relevant metadata for each reading. To retrieve past readings, the application accessed the IPFS network again through the client. It supplied a previously stored CID to the client's get method. This retrieved the full content file from nodes across the distributed system. By building upon the IPFS client API in this fashion, our mediator application conveniently integrated the privacy-preserving archival of sensitive IoT data streams within the decentralized IPFS framework.

Figure 9 conceptually depicts the publishing and retrieval workflows between the app and local node using the client. It helps illustrate how sensor payloads were anonymously pinned and later retrieved via their content-addressed unique identifiers.

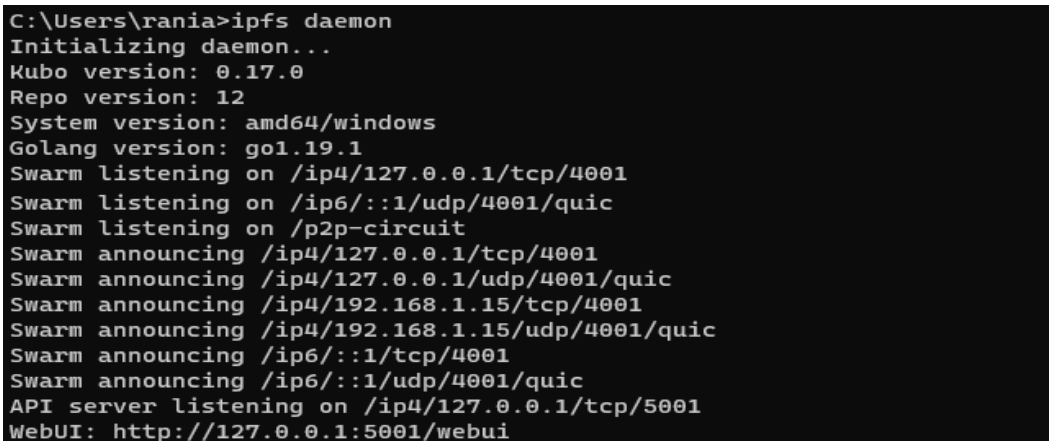

Figure 9. IPFS server startup.

It was now possible to continue using this CID to access the data via the IPFS, or to share it with other groups. Indeed, the uniquely generated CID during each addition to the IPFS constituted a permanent handle to indefinitely retrieve the associated content, regardless of its physical location on the network. With this CID, our application was therefore able to retrieve at any time the corresponding data stored in a decentralized manner. It was enough to call the IPFS client's "get" method by providing it with this CID. In addition, this same CID could be transparently shared with authorized third parties, so that they too could directly access the measurements on IPFS

via their own client. This demonstrated the interest of using a permanent and unique identifier like the CID to durably and share reference contents stored in a distributed manner on IPFS. Thanks to this key property of the CID, our solution provided real added value in terms of interoperability and long-term accessibility of IoT data published anonymously and decentralized.

While the IPFS provides a useful mechanism for distributing and addressing decentralized file contents, it does not inherently guarantee the immutability or permanence of those files over time. Specifically, an IPFS generates CIDs based on the cryptographic hash of the file data itself.

However, if the underlying file contents are ever modified or corrupted on the network, the hash and corresponding CID will change as well. This means the original version of the file can become effectively lost or inaccessible using its initial CID. To address this limitation, implementing mechanisms for permanently storing file metadata like CIDs can help preserve the integrity and traceability of archived contents on an IPFS. One approach is to leverage blockchain technologies, which provide an append-only distributed ledger ideal for recording immutable records. By saving each new CID generated when content is pinned to an IPFS on an underlying blockchain, we gain the ability to verify the provenance and authenticity of those files indefinitely into the future. Even if the files themselves were somehow altered at the storage layer of IPFS, the original metadata commitments preserved on-chain could be used to retrieve or reconstruct the initial versions. Projects like Filecoin take this synergy between IPFS and blockchains a step further by building dedicated protocols and economic incentives directly into their networks. Their services focus on issues like data replication, availability of offline/archived files and dispute resolution, helping to maximize the longevity and integrity of files referenced through CIDs on an IPFS. This offers a more robust solution than relying solely on IPFS alone for permanent, censorship-resistant storage and distribution of sensitive archival data over the long term.

4.3. Combining IPFS with Blockchain in New Irrigation System

Integrating IPFS into blockchain systems allows combining the benefits of these two technologies to efficiently and securely store and distribute data. By combining these two technologies, it is possible to store data efficiently while guaranteeing its integrity and confidentiality [48]. Specifically, IPFS provides an efficient decentralized storage and distribution solution through its distributed peer-to-peer network. However, it does not inherently ensure the immutability and verifiability of file contents over the long term.

Blockchain technology addresses this limitation by allowing the storage of metadata such as IPFS content

identifiers in an immutable, decentralized ledger. By pinning CIDs to the blockchain, their integrity and ability to trace files back to their original versions are permanently preserved. Even if files stored on an IPFS become corrupted or altered, their referenced metadata commitments on the blockchain can be used to detect issues and retrieve initial content. Together, the IPFS and blockchain create a robust archival solution. The IPFS efficiently stores and distributes file data through its global peer network. The immutable blockchain simultaneously anchors critical metadata like CIDs, cryptographic hashes and metadata necessary to validate file integrity and provenance in a tamper-proof way. This synergistic combination maximizes data integrity, permanence, confidentiality and censorship-resistance over the long run compared to each system alone.

We used MetaMask to send and receive transactions on the Ethereum network, as well as to interact with smart contracts. We used the Goerli test network for this work. MetaMask is a popular Ethereum wallet that runs as a browser extension. It allows users to easily connect to Ethereum-based blockchains like Goerli from websites without having to install full node software. By integrating with MetaMask, we were able to test functionality that involved blockchain transactions and contract calls directly from our application interface. MetaMask signs transactions and handles encryption/decryption behind the scenes [49]. The Goerli testnet provided a suitable sandbox environment for experimenting without using real Ether. It maintains the same protocols and consensus rules as the Ethereum mainnet but uses test coins so there is no monetary risk involved. Leveraging MetaMask and a test network allowed us to validate our approach for integrating an IPFS data pinning and archiving with the Ethereum blockchain in a safe, cost-effective manner before deploying to mainnet. It is crucial to carefully evaluate the specific requirements and tradeoffs related to our use case when managing near real-time data on the Ethereum blockchain. Considerations such as transaction costs, confirmation times, scalability and the need for decentralization should guide our decision-making process to find the optimal solution.

This step introduces smart contract development using Solidity and Remix IDE. Smart contracts are computer programs that automatically execute transactions and store data on a blockchain based on predefined rules and conditions. They provide a trusted way to transact digitally without an intermediary. To create smart contracts, developers typically use a specialized programming language designed for blockchain implementation. For Ethereum, the most popular language is Solidity. Solidity code is compiled down to bytecode that can be deployed onto the Ethereum blockchain and run on the Ethereum Virtual Machine (EVM). Remix IDE is an online tool

that facilitates the entire smart contract development process for Ethereum. It offers a web-based editor, compiler, test framework and debugger environment directly in the browser. These development tools make it easy to write, test and deploy Solidity contracts to local blockchain emulators like Ganache or public test networks. We will leverage Remix IDE to develop our first basic smart contract. Remix provides a contained space to quickly prototype and experiment with a smart contract code without risk. We will create a new .sol file to write the Solidity code, compile it, interactively test transactions and ultimately deploy the compiled bytecode to the Ethereum network for execution. This hands-on process introduces the overall workflow for smart contract engineering on Ethereum. By using Remix IDE, developers can efficiently explore solidity programming and smart contract functionality without requiring local infrastructure or transaction fees on the mainnet during the development cycle.

We begin by declaring a contract and specifying the Solidity version we want to use. Solidity versions help ensure compatibility and allow the compiler to identify compatible language features. Next, we define a variable called “CID” that will store the content identifier from our IPFS system. As IPFS generates unique CIDs for files added to its distributed network, we need a way for the smart contract to reference the stored content. Declaring the "CID" variable provides a field where the IPFS CID can be saved on-chain. This allows the contract to permanently pin the content to the blockchain by its identity, even if the underlying IPFS storage changes over time.

The Smart Contract employed to store CIDs on-chain would typically involve the following components:

- Contract Structure: The Smart Contract would define the necessary data structures and functions to store and retrieve CIDs. It would include variables or mappings to store the CIDs, along with associated metadata like timestamps or ownership information. The functions would enable adding, updating or retrieving CIDs from the contract.
- CID Storage: The CIDs themselves would be stored as part of the contract's state variables or mappings. The exact format and storage mechanism would depend on the implementation and requirements of the specific use case.
- Access Control: The Smart Contract would implement access control mechanisms to ensure that only authorized parties can add or modify CIDs. This could involve the use of permissioning schemes, such as role-based access control or whitelisting.

Deployment costs and transaction costs on the Ethereum blockchain can vary depending on several factors: network

congestion, optimization techniques, etc.

Regarding the impact of such an architectural solution in terms of latency and costs in a real-world scenario, it is indeed a context-dependent question that requires experimentation. The specific workload, data volume, sample rates and other factors would influence the latency and costs associated with storing CIDs on-chain. Conducting experiments within the specific context and domain of the application would provide valuable insights into the actual impact and feasibility of the solution. It is important to consider the unique requirements and constraints of the use case in question when evaluating the architectural solution, as they will greatly influence the latency, costs and overall viability of the approach.

The following figure (Figure 10) shows the opened smart contract deployment interface. We can choose the “injected provider MetaMask” deployment environment, the account to use for deployment, and define the contract parameters. After verifying that everything is correct, we can click "Deploy" to start the deployment process. Once deployment is complete, we can see the details of the deployed contract, including its address on the blockchain and available functions. We can also interact with the contract by using the available functions and sending transactions to it. Deploying the smart contract onto the blockchain makes it executable and accessible to all participants in the network in a decentralized manner. The deployment process involves compiling the Solidity code into EVM bytecode, estimating gas costs and broadcasting the signed deployment transaction through MetaMask to finalize on the network. The contract address links the bytecode to an account on Ethereum. This makes the application logic and data storage permanently accessible to anyone able to call the publicly available functions. We can now test running transactions through the contract to validate its operation before making it available for live use on the mainnet.

Figure 10. Smart contract deployment

When the smart contract is successfully deployed, the Remix IDE will display a confirmation message and provide information about the contract (see Figure 11), such as the contract address and the deployment cost in ETH. The successful deployment of the contract onto the blockchain makes it executable and accessible in a decentralized manner to all network participants. Remix IDE then displays key details like the permanent contract address assigned on the Ethereum blockchain. This allows it to be uniquely identified.

Figure 11. Deployment result of a smart contract.

The IDE also provides the total cost incurred for deployment, expressed in Ether (ETH). This includes the gas fees required to process and validate the transaction on the network. This information is useful to verify that the deployment process went smoothly and have the necessary references to then interact with the contract directly on the blockchain. With the feedback from Remix IDE, the developer can ensure the contract was

deployed to the intended address before starting to test the programmed functionality. The contract address and deployment transaction details allow interaction and verification of the smart contract as intended.

After completing the implementation of our application, we can now transfer the data to the blockchain.. To further verify the proper functioning of our application, we used the retrieval function “Contract.Function.getTTN()” which displays the same CID values that we sent to the blockchain. Successfully storing the CID on-chain and then retrieving it validates that the contract is working as intended to pin content references permanently. Displaying the uploaded CID and its matching hashed value provides a checkpoint that the key application stages are occurring properly:

- CID generation when content is added to IPFS.
- CID submission to smart contract.
- Retrieval of stored CID from smart contract.

Seeing the original and retrieved CID match gives confidence that the IPFS–blockchain integration is functioning end-to-end to archive metadata immutably on the distributed ledger. This final test retrieval step reinforces that the overall application logic and flow was implemented correctly.

Finally, we can see using Etherscan the blocks and transactions between the IPFS and the blockchain. We can see that these transactions are encrypted as mentioned, and can only be decrypted by the smart contract using the retrieval function that we developed. Being able to view the on-chain activity provides validation that the IPFS–blockchain integration is working securely. Etherscan allows exploring the specific blocks, transactions and contract status over time. We can observe that the transactions sent from our application to pin the content metadata references are properly encrypted. Only our smart contract's retrieval function holds the logic to decrypt these transactions and access the underlying CID values, demonstrating data confidentiality. Seeing the encrypted transaction flow corroborates our approach of chaining IPFS content identities to the blockchain through cryptographic bindings that are decipherable solely by the contract. This final monitoring step using Etherscan's transaction explorer proves that the data transfer and pinning mechanism was built securely as designed.

4.4. Discussion and Advantages

The results of this study demonstrate the potential of integrating IoT, IPFS and blockchain technologies to address data security and management challenges for IoT systems. When compared to existing approaches, some key differences

and contributions of this research are:

- Decentralized Data Management: Unlike centralized database solutions, the use of an IPFS for distributed storage ensures data availability even without single points of failure. This provides higher resilience than studies relying solely on cloud or edge computing.
- Immutable Recordkeeping: The integration of blockchain differs from prior works utilizing only cloud-based or traditional distributed databases by providing an append-only transaction ledger with tamper-proof recordkeeping abilities not found in other systems.
- Automated Processes: By implementing smart contracts, this research expands upon rule-based IoT system architectures by introducing a method for secure, self-executing process automation not seen in rule engine-driven approaches.
- Transparent Auditing: The combination of IPFS hashes and blockchain transactions incorporated here distinguishes this work from others using encryption-only or private ledger schemes by facilitating public verifiability and auditing of data and system activity.
- Practical Evaluation: Unlike conceptual Architecture-only studies, this research contributes a validated implementation and performance analysis of an integrated IoT–IPFS–blockchain system for a real-world automatic irrigation use case.

In summary, this work provides both theoretical and methodological innovations through the design, implementation and evaluation of a decentralized, trustworthy architecture leveraging synergistic abilities of emerging technologies in a novel combination differentiated from prior related research. The findings offer practical guidance for improving IoT security and operations through distributed technologies.

5. Conclusion and Perspectives

In this paper, we developed an automated and secure irrigation system by integrating the innovative blockchain and IPFS technologies. The practical tests demonstrated the proper functioning of our proposed solution. The key aspects we addressed were:

- Immutably storing irrigation schedule and sensor data on the blockchain for tamper-proof record keeping.
- Leveraging IPFS to durably archive sensor logs and irrigation logs in a distributed file system.
- Autonomously executing irrigation commands on schedule or based on sensor triggers via a smart contract.
- Providing a dashboard interface to view data and manage the system in a decentralized manner.

The end-to-end implementation proved the validity of pinning IPFS content backups to the blockchain using

cryptographic bindings like CIDs. Storing metadata on-chain with recallable off-chain data preserves integrity over the long run in a resilient design. Going forward, areas for improvement include enhancing the dashboard, optimizing storage usage and adding automated analytics/reports. Potential new features involve predicting maintenance needs, improving efficiency and integrating additional IoT sensors for expanded monitoring. This work showcased how integrating blockchain and IPFS can deliver reliable, secure and trustless solutions for real-world decentralized applications like smart agriculture systems. The approach holds promise for broader adoption in industrial domains.

There is significant potential to further develop and scale this integrated blockchain–IPFS solution for smart agriculture use cases. Broader adoption could see deployment on larger farms with expanded sensor networks monitoring more fields, crops and irrigation zones. Over time, standardizing communication protocols would facilitate interoperability with diverse IoT equipment. Capturing and sharing harvest and production data on the blockchain from farm to consumer could also enable full supply chain traceability.

Leveraging the growing volumes of sensor information, machine learning algorithms could help optimize irrigation scheduling and maximize water and energy efficiency. This would support sustainability goals while improving yields. Additionally, deploying renewable energy technologies like solar power on agricultural lands could reduce dependence on electricity grids. The blockchain layer could further harden security by protecting sensor data against hacking or tampering. New data-driven business models may emerge, such as offering analytics services or crop insurance programs based on verifiable on-chain production records. As the ecosystem matures, streamlining storage usage and enabling automated reporting would enhance usability. With continued innovation, the vision of global smart farming powered by decentralized technologies could move closer to realization, benefiting producers and consumers alike.

References

Rejeb, A.; Rejeb, K.; Zailani, S.H.M.; Abdollahi, A. Knowledge Diffusion of the Internet of Things (IoT): A Main Path Analysis. Wirel. Pers. Commun. 2022, 126, 1177–1207.

https://doi.org/10.1007/s11277-022-09787-8.

Partha, R.; Neeraj, K. SDN/NFV architectures for edge-cloud oriented IoT: A systematic review. Comput. Commun. 2021, 169, 129–153. https://doi.org/10.1016/j.comcom.2021.01.018.

Sangeeta, N.; Nam, S.Y. Blockchain and Interplanetary File System (IPFS)-Based Data Storage System for Vehicular Networks with

Keyword Search Capability. Electronics 2023, 12, 1545. https://doi.org/10.3390/electronics12071545.

Smita, A.; Ramesh, T. Blockchain based hierarchical semi-decentralized approach using IPFS for secure and efficient data sharing. J. King Saud Univ. - Comput. Inf. Sci. 2022, 34, 1523–1534. https://doi.org/10.1016/j.jksuci.2022.01.019.

Gugueoth, V.; Sunitha, S.; Sachin, S.; Danda, R. A review of IoT security and privacy using decentralized blockchain techniques. Comput. Sci. Rev. 2023, 50, 100585. https://doi.org/10.1016/j.cosrev.2023.100585.

Sultana, S.A.; Rupa, C.; Malleswari, R.P.; Gadekallu, T.R. IPFS-Blockchain Smart Contracts Based Conceptual Framework to Reduce Certificate Frauds in the Academic Field. Information 2023, 14, 446. https://doi.org/10.3390/info14080446.

Ashok, K.; Gopikrishnan, S. Statistical Analysis of Remote Health Monitoring Based IoT Security Models & Deployments from a Pragmatic Perspective. IEEE Access 2023, 11, 2621–2651. https://doi.org/10.1109/ACCESS.2023.3234632.

Alaya, B. Efficient privacy-preservation scheme for securing urban P2P VANET networks. Egypt. Inform. J. 2021, 22, 317–328. https://doi.org/10.1016/j.eij.2020.12.002.

Dhanaraju, M.; Chenniappan, P.; Ramalingam, K.; Pazhanivelan, S.; Kaliaperumal, R. Smart Farming: Internet of Things (IoT)-Based Sustainable Agriculture. Agriculture 2022, 12, 1745. https://doi.org/10.3390/agriculture12101745.

Bashynska, I.; Mukhamejanuly, S.; Malynovska, Y.; Bortnikova, M.; Saiensus, M.; Malynovskyy, Y. Assessing the Outcomes of Digital Transformation Smartization Projects in Industrial Enterprises: A Model for Enabling Sustainability. Sustainability 2023, 15, 14075. https://doi.org/10.3390/su151914075.

Senthilkumar, M.; Akshaya, R.; Sreejith, K. An Internet of Things-based Efficient Solution for Smart Farming. Procedia Comput. Sci. 2023, 218, 2806–2819.

https://doi.org/10.1016/j.procs.2023.01.252.

Badran, A.I.; Kashmoola, M.Y. Smart agriculture using Internet of Things: A Survey. In Proceedings of the 1st International Multi-Disciplinary Conference Theme: Sustainable Development and Smart Planning, IMDC-SDSP 2020, Cyperspace, 28–30 June 2020; p. 10.

Kim, T.-h.; Solanki, V.S.; Baraiya, H.J.; Mitra, A.; Shah, H.; Roy, S. A smart, sensible agriculture system using the exponential moving average model. Symmetry 2020, 12, 457.

Ramli, M.R.; Ramli, P.; Daely, T.; Kim, D.-S.; Lee, J.M. IoT-based adaptive network mechanism for reliable smart farm system. Comput. Electron. Agric. 2020, 170, 105287.

Raju, K.L.; Vijayaraghavan, V. IoT technologies in agricultural

environment: A survey. Wirel. Pers. Commun. 2020, 113, 2415–2446.

Akhigbe, B.I.; Munir, K.; Akinade, O.; Akanbi, L.; Oyedele, L.O. IoT Technologies for Livestock Management: A Review of Present Status, Opportunities, and Future Trends. Big Data Cogn. Comput. 2021, 5, 10. https://doi.org/10.3390/bdcc5010010.

Kamalendu, P.; Ansar, U.H.Y. Internet of Things Impact on Supply Chain Management. Procedia Comput. Sci. 2023, 220, 478–485. https://doi.org/10.1016/j.procs.2023.03.061.

Kim, S.; Yoo, Y. SLSMP: Time synchronization and localizationusing seawater movement pattern in underwater wireless networks. Int.J. Distrib. Sens. Netw. 2014, 10, 172043.

Stoianova, I.; Stoianov, L. Nachman, S. Madden, and T. Tokmouline. PIPENETawireless sensor network for pipeline monitoring. In Proceedings of the 2007 6th International Symposium on Information Processing in Sensor Networks (IPSN), Cambridge, MA, USA, 25–27 April 2007, pp. 264–273.

Grace, R.K.; Manju, S. A comprehensive review of wireless sensornetworks based air pollution monitoring systems. Wirel. Pers. Commun. 2019, 108, 2499–2515.

Chen, D.; Liu, Z.; Wang, L.; Dou, M.; Chen, J.; Li, H. Natural disastermonitoring with wireless sensor networks: A case study of data-intensiveapplications upon low-cost scalable systems. Mob. Netw. Appl. 2013, 18, 651–663.

Ellouze, N.; Rekhis, S.; Boudriga, N. A WSN-based solution forpollution detection and localization in waterways. Arab. J. Sci. Eng. 2019, 44, 3213–3233.

Gayal, N.; Dave, M.; Verma, A.K. Protocol stack of underwaterwireless sensor network: Classical approaches and new trends. Wireless Pers. Commun. 2019, 104, 995–1022.

Hamdan, S.; Ayyash, M.; Almajali, S. Edge-Computing Architectures for Internet of Things Applications: A Survey. Sensors 2020, 20, 6441. https://doi.org/10.3390/s20226441.

Xiong, Z.; Zhang, Y.; Luong, N.C.; Niyato, D.; Wang, P.; Guizani, N. The best of both worlds: A general architecture for data management in blockchain-enabled internet-of-things. IEEE Netw. 2020, 34, 166–173.

Ali, M.S.; Dolui, K.; Antonelli, F. Iot data privacy via blockchains and ipfs. In Proceedings of the Seventh International Conference on the Internet of Things, At: Linz, Austria, 2017; pp. 1–7.

Dai, H.-N.; Zheng, Z.; Zhang, Y. Blockchain for internet of things: A survey. IEEE Internet Things J. 2019, 6, 8076–8094.

Wang, X.; Zha, X.; Ni, W.; Liu, R.P.; Guo, Y.J.; Niu, X.; Zheng, K. Survey on blockchain for internet of things. Comput. Commun. 2019, 136, 10–29.

Theoleyre, F.; Pang, A.-C. Internet of Things and M2M

Communications; River Publishers: Aalborg, Denmark, 2013.

Chen, S.; Xu, H.; Liu, D.; Hu, B.; Wang, H. A vision of iot: Applications, challenges, and opportunities with China perspective. IEEE Internet Things J. 2014, 1, 349–359.

Alam, M.; Ferreira, J.; Mumtaz, S.; Jan, M.A.; Rebelo, R.; Fonseca, J.A. Smart cameras are making our beaches safer: A 5g-envisioned distributed architecture for safe, connected coastal areas. IEEE Veh. Technol. Mag. 2017, 12, 50–59.

Minoli, D.; Occhiogrosso, B. Blockchain mechanisms for iot security. Internet Things 2018, 1, 1–13.

Makhdoom, I.; Abolhasan, M.; Abbas, H.; Ni, W. Blockchain's adoption in iot: The challenges, and a way forward. J. Netw. Comput. Appl. 2019, 125, 251–279.

Kshetri, N. Can blockchain strengthen the internet of things? IT Prof. 2017, 19, 68–72.

Dinh, T.T.A.; Liu, R.; Zhang, M.; Chen, G.; Ooi, B.C.; Wang, J. Untangling blockchain: A data processing view of blockchain systems. IEEE Trans. Knowl. Data Eng. 2018, 30, 1366–1385.

Khan, M.A.; Salah, K. Iot security: Review, blockchain solutions, and open challenges. Future Generat. Comput. Syst. 2018, 82, 395–411.

Swan, M. Blockchain: Blueprint for a New Economy; O'Reilly Media, Inc.: Sebastopol, CA, USA, 2015. Taivalsaari, A.; Mikkonen, T. A roadmap to the programmable world: Software challenges in the iot era. IEEE Softw. 2017, 34, 72–80.

Reyna, A.; Martn, C.; Chen, J.; Soler, E.; Daz, M. On blockchain and its integration with iot. Challenges and opportunities. Future Generat. Comput. Syst. 2018, 88, 173–190.

Alaya, B.; Khan, R.; Moulahi, T.; Khediri, S. Study on QoS Management for Video Streaming in Vehicular Ad Hoc Network (VANET). Wirel. Pers. Commun. 2021, 118, 2175–2207. https://doi.org/10.1007/s11277-021-08118-7.

Kuo, T.T.; Kim, H.-E.; Ohno-Machado, L. Blockchain distributed ledger technologies for biomedical and health care applications. J. Am. Med. Inf. Assoc. 2017, 24, 1211–1220.

Hackius, N.; Petersen, M. Blockchain in logistics and supply chain: Trick or treat? In Proceedings of the Hamburg International Conference of Logistics (HICL), Berlin, Germany, 2017; pp. 3–18.

Lee, J.; Mtibaa, A.; Mastorakis, S. A case for compute reuse in future edge systems: An empirical study. In Proceedings of the 2019 IEEE Globecom Workshops (GC Wkshps), Waikoloa, HI, USA, 9–13 December 2019; pp. 1–6.

Salah, K.; Rehman, M.H.U.; Nizamuddin, N.; Al-Fuqaha, A. Blockchain for ai: Review and open research challenges. IEEE Access 2019, 7, 10127–10149.

Viriyasitavat, W.; Da Xu, L.; Bi, Z.; Hoonsopon, D.; Charoenruk, N.

Managing qos of internet-of-things services using blockchain. IEEE Trans. Comput. Soc. Syst. 2019, 6, 1357–1368.

Bashynska, I.; Malanchuk, M.; Zhuravel, O.; Olinichenko, K. Smart Solutions: Risk Management of Crypto-Assets and Blockchain Technology. Int. J. Civ. Eng. Technol. 2019, 10, 1121–1131.

Alzahrani, A.O.; Alenazi, M.J.F. Designing a Network Intrusion Detection SystemBased on Machine Learning for Software Defined Networks. Future Internet 2021, 13, 111. https://doi.org/10.3390/fi 13050111.

Sellami, L.; Alaya, B. UPSO-FSVRNET: Fuzzy Identification Approach in a VANET Environment Based on Fuzzy Support Vector Regression and Unified Particle Swarm Optimization. Int. J. Fuzzy Syst. 2022, 25, 743–762.

https://doi.org/10.1007/s40815-022-01408-7.

Stephan, B.; Olga, B.; Röglinger, M. Attacks on the Industrial Internet of Things—Development of a multi-layer Taxonomy. Comput. Secur. 2020, 93, 101790.

https://doi.org/10.1016/j.cose.2020.101790.

Alaya, B.; Laouamer, L.; Msilini, N. Homomorphic encryption systems statement: Trends and challenges. Comput. Sci. Rev. 2020, 36, 100235. https://doi.org/10.1016/j.cosrev.2020.100235.