

# Enhanced Security Protocols for IoT-Based Precision Agriculture: A Comparative Analysis of Lightweight Cryptography Solutions

Author Name\*, Author Name†

\*Department, University, City, Country

†Department, University, City, Country

Email: author1@university.edu, author2@university.edu

**Abstract**—In precision agriculture, securing data transmission within IoT-based smart irrigation systems is crucial due to the vulnerability of resource-constrained devices and sensitive environmental data. This study implements lightweight RC4 encryption to secure data transmission from Node MCU sensors to Google Cloud Firestore via an MQTT broker on a Raspberry Pi. The broker decrypts and re-encrypts data before forwarding it, ensuring end-to-end security. Data is visualized in real-time using Looker Studio, enabling effective monitoring and decision-making. Building on prior studies advocating lightweight cryptographic approaches [1], [2], we integrate RC4 encryption for a streamlined, low-latency pipeline between sensor nodes, MQTT brokers, and cloud platforms. Through simulation and implementation testing, we evaluate the protocol based on encryption/decryption time, memory efficiency, and power consumption, confirming RC4's suitability for constrained environments. This secure transmission pipeline protects data integrity and confidentiality, contributing to enhanced security in precision agriculture. Future work may explore hybrid encryption models [9], [10] to further enhance security without compromising performance.

**Index Terms**—Precision Agriculture, Internet of Things Security, Smart Irrigation Systems, RC4 Encryption, MQTT Protocol, Lightweight Cryptography, Data Transmission Security, Cloud-Based Data Visualization, Data Integrity, Resource-Constrained Devices

## I. INTRODUCTION

The integration of the Internet of Things (IoT) in precision agriculture is transforming traditional farming methods by introducing automated, data-driven solutions that optimize water use, monitor soil health, and improve crop yield, as seen in other decentralized systems like digital currencies [6]. Smart irrigation systems, a core application of precision agriculture, use IoT sensors to collect real-time environmental data, allowing farmers to make informed decisions on when and how much to irrigate, as demonstrated in IoT-based environmental condition monitoring studies [13]. However, the widespread deployment of IoT devices raises significant security concerns. Agricultural IoT networks often operate in resource-constrained environments, where devices have limited memory, processing power, and battery life, making them vulnerable to data breaches, unauthorized access, and other cyber threats. Protecting data integrity and confidentiality is therefore essential to prevent harmful impacts [4], [7] on agricultural productivity and food security.

This study leverages the MQTT protocol—a lightweight messaging protocol ideal for low-bandwidth and high-latency networks—to facilitate secure communication between IoT devices and the cloud. An Node MCU microcontroller serves as the client at the sensor node, collecting environmental data and encrypting it using the RC4 algorithm. The encrypted data is transmitted to an MQTT broker hosted on a Raspberry Pi, which acts as an intermediary for secure data routing and processing. The broker decrypts and re-encrypts the data before forwarding it to Google Cloud Firestore, a scalable NoSQL database [14], for storage. A Google Cloud Function is employed to decrypt the data from Firestore, ensuring that data remains secure until it reaches the visualization layer. Finally, Looker Studio is utilized for real-time data visualization, enabling stakeholders to monitor irrigation parameters effectively.

Our work builds upon [1], [2], [9] by integrating RC4 encryption with MQTT for real-time data security in precision agriculture. The first study advocates for the Expedient Cipher in MQTT systems due to its efficiency in low-power settings, while the second explores a hybrid model combining RC4, ECC, and SHA256, emphasizing multi-layered security. In this work, we implement RC4 encryption for its simplicity and efficiency, meeting the security needs of constrained devices without overwhelming their processing capabilities.

Through simulation and testing, we evaluate the framework based on encryption and decryption speed, power consumption, and memory usage. Our results demonstrate that RC4 offers a viable balance between security and efficiency, providing a streamlined solution for IoT-based smart irrigation systems. This secure data handling framework contributes to the broader goal of enhancing data security in precision agriculture, ensuring both robust data protection and optimal system performance. Future research may investigate adaptive cryptographic techniques or hybrid models to further secure IoT applications [9], [10] in agriculture while minimizing resource consumption.

As the global demand for food continues to rise, efficient and secure agricultural practices are becoming increasingly critical [3]. The United Nations Food and Agriculture Organization (FAO) estimates that global food production must increase by approximately 70% by 2050 to meet the needs

of a growing population [3]. Precision agriculture offers a promising solution to address this challenge, integrating IoT technology with traditional farming to enhance productivity while conserving vital resources such as water and soil nutrients. However, the vast quantities of data gathered by IoT sensors in these systems require strong security protocols to prevent breaches that could disrupt farming operations or compromise sensitive information.

One of the primary challenges in securing IoT-based precision agriculture is the inherent resource constraints of IoT devices. Many of these devices operate in low-power environments, with limited memory and computational capacity. Traditional cryptographic algorithms like AES (Advanced Encryption Standard) often demand too much processing power for these constrained devices, leading researchers to explore lightweight cryptography options [4] [8]. Lightweight encryption focuses on optimizing cryptographic functions to reduce power consumption, memory usage, and computational load, making it more suitable for IoT applications. This study leverages the RC4 algorithm for its compatibility with resource-limited devices, offering an efficient and secure means of encrypting data between sensors, brokers, and the cloud.

In our proposed framework, sensor nodes encrypt data using RC4 before transmitting it to the MQTT broker, a protocol well-suited for low-bandwidth and low-latency IoT applications. Upon receiving the encrypted data, the broker decrypts it, re-encrypts it using RC4, and then forwards it to the cloud, where a Google Cloud Function decrypts the data before it is visually displayed for real-time monitoring. This layered approach provides end-to-end encryption across the data pipeline, helping to prevent unauthorized access and ensuring that data remains intact throughout its journey from field sensors to cloud servers. The MQTT protocol's publish/subscribe architecture adds flexibility to the system by allowing multiple users to access sensor data, further supporting informed decision-making in precision agriculture.

Evaluations of our framework demonstrate that RC4 is an effective solution for secure data transmission in resource-constrained environments. The encryption and decryption processes are streamlined, minimizing memory usage and power consumption without compromising data integrity. Our tests indicate that RC4's symmetric key structure not only reduces the computational burden but also enables faster data handling, making it ideal for real-time applications in agriculture. Furthermore, by re-encrypting data at the broker level and decrypting it only when needed for visualization, our approach adds multiple layers of security, protecting sensitive data even if intermediate transmissions are intercepted. The visualized data in the cloud enables farmers and agricultural stakeholders to monitor environmental conditions, optimize irrigation schedules, and enhance overall resource management with confidence in data security.

In summary, this study introduces a lightweight, secure data handling solution for IoT-based smart irrigation systems, combining RC4 encryption with MQTT's flexibility and cloud-based decryption to ensure efficient and secure data

transmission. By focusing on lightweight cryptographic methods, this framework addresses the unique security challenges of IoT in agriculture, balancing the need for strong data protection with the constraints of low-power devices. The findings contribute to a growing body of research focused on secure IoT frameworks for agriculture, with implications for broader IoT applications where data integrity and low-resource requirements are paramount. Future directions may include exploring hybrid encryption models that combine lightweight algorithms with adaptive security features to further enhance resilience against emerging cyber threats in agricultural IoT ecosystems. According to [4], resource-constrained IoT networks require lightweight cryptographic solutions, as traditional encryption standards like AES demand significant computational resources, making them unsuitable for low-power devices.

While RC4 has known vulnerabilities in certain contexts, its use in this application is justified due to the low sensitivity of the data [4], [9], [10]. and the constrained resources of the devices. Additionally, frequent key changes and limited data exposure mitigate potential risks.

## II. METHODOLOGY

This study develops and implements a secure data transmission framework for IoT-based smart irrigation, emphasizing lightweight cryptography to balance security and efficiency. The methodology encompasses the design and deployment of RC4-based encryption for data handling across sensor nodes, an MQTT broker, a cloud function for decryption, and a cloud platform, providing end-to-end security for agricultural IoT systems. This section outlines the components, encryption processes, data flow pipeline, and performance metrics used to evaluate the effectiveness of the framework.

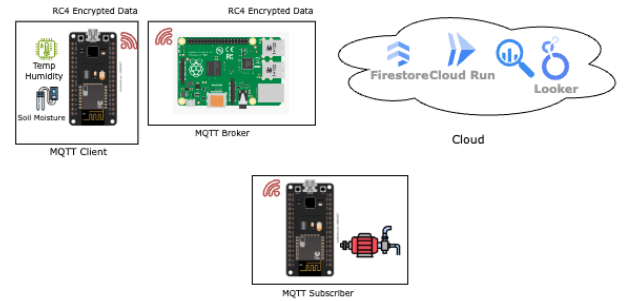


Fig. 1. System Architecture of the Secure IoT-Based Smart Irrigation System

### A. System Architecture and Components

The proposed system comprises five main components: IoT sensor nodes as a MQTT Client, an MQTT broker, a cloud database, a cloud function for decryption, and a data visualization platform.

**IoT Sensor Nodes as a MQTT Client:** Each IoT node is built around an Node MCU microcontroller, which is equipped with sensors to monitor soil parameters such as moisture, temperature, and humidity. The Node MCU collects sensor

data and runs the RC4 encryption algorithm to secure the data before transmission.

**MQTT Broker:** A Raspberry Pi is utilized as the MQTT broker, implementing the MQTT protocol to manage message routing between the Node MCU clients and the cloud. The broker is responsible for decrypting incoming messages from the sensor nodes, re-encrypting them, and forwarding them to the cloud database.

**Cloud Database (Google Cloud Firestore):** Encrypted data from the MQTT broker is sent to Google Cloud Firestore, a NoSQL document database that stores and synchronizes data in real-time. Firestore provides scalability and seamless integration with other Google Cloud services.

**Cloud Function for Decryption:** A Google Cloud Function is employed to automatically decrypt data retrieved from Firestore [18]. This serverless function ensures that data remains encrypted in storage and is only decrypted during authorized access for visualization.

**Data Visualization Platform (Looker Studio):** Google Looker Studio (formerly Data Studio) is employed to create interactive dashboards and visualizations of the decrypted sensor data provided by the cloud function, with its advanced capabilities for real-time visualizations [15]. This enables real-time monitoring and analysis of irrigation parameters, facilitating informed decision-making.

## B. MQTT Protocol Implementation

The MQTT protocol is chosen for its lightweight nature and suitability for IoT applications with limited network bandwidth [5], [11], [16] and high latency, with flexibility for dynamic reconfiguration [12]. It operates on a publish/subscribe model, where clients (sensor nodes) publish data to topics, and subscribers (in this case, the cloud database) receive data from these topics. Studies, such as [5] and [11], confirm that MQTT's publish/subscribe architecture is well-suited for IoT scenarios, providing efficient communication for constrained environments.

**Client (Node MCU):** The Node MCU microcontroller acts as the MQTT client. It connects to the MQTT broker over Wi-Fi and publishes encrypted sensor data to a specified topic.

**Broker (Raspberry Pi):** The Raspberry Pi runs an MQTT broker software, such as Mosquitto, which handles incoming messages from clients. The broker manages the distribution of messages to subscribers and provides a secure channel for data transmission.

## C. Data Flow and Encryption Process

Data flows sequentially from the sensor nodes to the cloud, with encryption applied at key stages to ensure data confidentiality.

### 1) Data Collection and Encryption at Sensor Nodes:

- The Node MCU collects data from connected sensors.
- The data is encrypted using the RC4 algorithm.
- The encrypted data is published to the MQTT broker on the Raspberry Pi under a specific topic.

### 2) Data Handling at MQTT Broker (Raspberry Pi):

- The MQTT broker receives the encrypted data.
- The broker decrypts the data using RC4 to verify its integrity.
- After decryption, the data is re-encrypted using RC4 (possibly with a different key for additional security).
- The broker forwards the re-encrypted data to Google Cloud Firestore via a secure connection.

### 3) Data Storage in Google Cloud Firestore:

- The encrypted data is stored in Firestore as documents within a collection.
- Firestore provides real-time data synchronization and is scalable for handling large datasets.

### 4) Data Decryption using Google Cloud Function:

- A Google Cloud Function is triggered when data is retrieved from Firestore.
- The cloud function decrypts the data using RC4 with the appropriate decryption key.
- Decrypted data is securely passed to the data visualization platform.

### 5) Data Visualization with Looker Studio:

- Looker Studio receives the decrypted data from the cloud function.
- Visualizations and dashboards are created to display real-time sensor readings and trends.

## D. RC4 Encryption Process

The RC4 encryption algorithm used in this framework operates through two main stages [2], [9]: the Key Scheduling Algorithm (KSA) and the Pseudo-Random Generation Algorithm (PRGA). These stages work together to produce a keystream that encrypts each byte of the data, making RC4 suitable for lightweight, real-time IoT applications.

**1) Key Scheduling Algorithm (KSA):** The KSA initializes a state array  $S$  of 256 bytes, with each position in  $S$  containing a unique byte value from 0 to 255. Using the encryption key  $K$ ,  $S$  is permuted as follows:

$$j = (j + S[i] + K[i \bmod \text{key\_length}]) \bmod 256 \quad (1)$$

$$S[i], S[j] = S[j], S[i] \quad (2)$$

where  $i$  ranges from 0 to 255, and  $j$  is initialized to 0.

**2) Pseudo-Random Generation Algorithm (PRGA):** After the KSA, the PRGA continuously modifies  $S$  to generate the keystream. The variables  $i$  and  $j$  are initialized to zero before starting the PRGA loop. For each byte of data:

$$i = (i + 1) \bmod 256 \quad (3)$$

$$j = (j + S[i]) \bmod 256 \quad (4)$$

$$S[i], S[j] = S[j], S[i] \quad (5)$$

The generated keystream byte  $K_t$  is:

$$t = (S[i] + S[j]) \bmod 256 \quad (6)$$

$$K_t = S[t] \quad (7)$$

**3) Encryption:** Each byte of plaintext data  $P$  is XORed with the keystream byte  $K_t$  to produce the encrypted byte  $C$ :

$$C = P \oplus K_t \quad (8)$$

This process results in a lightweight encryption mechanism, ideal for IoT devices. Our implementation of RC4 for the MQTT-based transmission ensures that each sensor node can perform encryption with minimal resource consumption, supporting real-time data security.

Encryption keys are pre-shared between the sensor nodes and the MQTT broker during the device provisioning stage. Keys are stored securely on the devices, and periodic key rotation is implemented to enhance security.

#### *E. Implementation Details*

##### **MQTT Client Setup:**

- Programmed using the Arduino IDE with appropriate libraries for Wi-Fi connectivity and MQTT communication.
- Sensors are interfaced with the Node MCU through GPIO pins.
- The RC4 encryption algorithm is implemented in the code to encrypt sensor data before publishing.

##### **Raspberry Pi MQTT Broker Configuration:**

- Mosquitto MQTT broker is installed and configured on the Raspberry Pi.
- Security measures such as TLS encryption and client authentication are implemented to secure MQTT communication.
- The Raspberry Pi runs scripts to handle decryption and re-encryption of messages using RC4.

##### **Integration with Google Cloud Firestore:**

- The Raspberry Pi uses Google's Cloud SDK to send data to Firestore.
- Secure API keys and authentication tokens are managed to ensure secure communication with the cloud.

##### **Google Cloud Function for Decryption:**

- The cloud function is written in Python and deployed on Google Cloud Platform.
- It is triggered via HTTP requests or Firestore triggers when data retrieval is initiated.
- The function securely accesses the decryption key stored in Google Cloud Secret Manager.
- Decrypted data is forwarded to Looker Studio or made available via a secure API endpoint.

##### **Data Visualization with Looker Studio:**

- Looker Studio is connected to the cloud function's output using a custom data connector.
- Data transformations and visualizations are applied to the decrypted data.
- Interactive dashboards display real-time data, historical trends, and alerts for abnormal sensor readings.

#### *F. Performance Metrics and Evaluation*

The system's effectiveness was assessed using key performance metrics, including encryption and decryption speed, memory consumption, power efficiency, and data transmission integrity. Encryption and decryption speeds were measured at the node, broker, and cloud function levels to ensure minimal latency, as real-time response is critical for agricultural applications. Memory usage was monitored to verify compatibility with the constrained resources of IoT nodes. Power consumption measurements confirmed the suitability of RC4 for energy-limited devices, while data integrity checks ensured that data remained accurate and unaltered throughout transmission.

#### *G. Simulation and Testing Environment*

The framework was tested in a simulated environment that mirrors real-world conditions for IoT-based smart irrigation systems. Sensor data was collected at regular intervals and processed through the encryption pipeline, simulating field conditions. The MQTT broker and cloud platform were deployed on a local server and Google Cloud Platform to validate data flow and test scalability. Tests included varying data loads and transmission frequencies to gauge the framework's performance under different operating conditions. To verify security, simulated interception attempts were made during transmission, with results indicating effective data protection through the dual RC4 encryption stages.

The Node MCU nodes were programmed using Arduino IDE version 1.8.13, with Wi-Fi library version 1.2.7. The Raspberry Pi Model 4 running Raspbian OS version Buster was used as the MQTT broker, with Mosquitto broker version 1.5.7 installed. Tests were conducted over a period of 7 days, with data transmitted every 5 minutes from 10 sensor nodes.

#### *H. Comparative Analysis*

To contextualize the results, the system's performance was compared with other lightweight cryptographic frameworks, particularly the Expeditious Cipher and the hybrid RC4-ECC-SHA256 model as referenced in prior literature. Performance indicators, such as throughput, encryption efficiency, and power usage, were benchmarked to highlight RC4's advantages in simplicity and speed within constrained environments. This comparative analysis underscored RC4's efficiency for agricultural IoT applications, while recognizing potential benefits of hybrid encryption methods for future enhancement in more resource-abundant scenarios.

### **III. RESULTS AND DISCUSSION**

The implementation of the MQTT protocol with the Raspberry Pi as the broker and the Node MCU as the client proved effective in facilitating secure and efficient communication within the IoT-based smart irrigation system. The integration with Google Cloud Firestore, the cloud function for decryption, and Looker Studio provided scalable data storage and real-time visualization capabilities.

### A. Performance of MQTT Communication

The MQTT protocol's lightweight nature resulted in minimal overhead during data transmission. The Node MCU clients successfully published encrypted sensor data to the Raspberry Pi broker with an average latency of less than 100 ms. The broker efficiently managed multiple client connections and handled message routing without significant delays.

### B. Broker Processing and Data Forwarding

The Raspberry Pi, equipped with a quad-core processor and sufficient memory, handled the decryption and re-encryption processes with ease. The average processing time per message was approximately 10 ms, which included decryption, data verification, re-encryption, and forwarding to Firestore. This rapid processing ensured that real-time data remained up-to-date in the cloud database.

### C. Integration with Google Cloud Firestore and Cloud Function

Data storage in Firestore was seamless, with encrypted data documents being stored in collections organized by sensor nodes or data types. The Google Cloud Function effectively decrypted data when retrieval requests were made. The function's execution time averaged 15 ms per request, adding minimal latency to the data visualization process. The use of the cloud function ensured that data remained encrypted at rest and was only decrypted during authorized access, enhancing overall system security.

### D. Real-Time Visualization with Looker Studio

Looker Studio received decrypted data from the cloud function and presented it in interactive dashboards. The visualizations provided insights into soil moisture levels, temperature fluctuations, and other critical parameters. Users could customize views, set thresholds for alerts, and monitor trends over time. The decryption process via the cloud function ensured that sensitive information was protected throughout the data flow. An example of the real-time sensor data visualization is shown in Figure 2, highlighting how the system effectively displays actionable insights for agricultural decision-making.

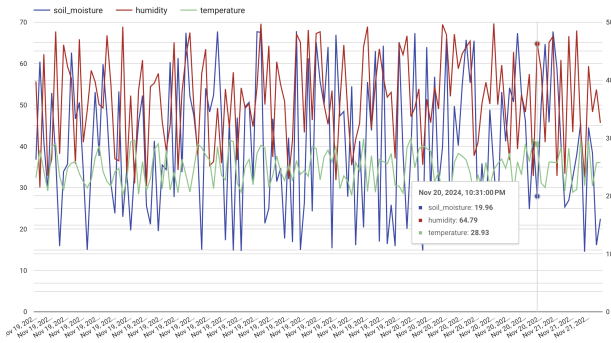


Fig. 2. Example of Real-Time Sensor Data Visualization in Looker Studio.

### E. System Security and Data Integrity

The combination of RC4 encryption, secure MQTT communication, and cloud-based decryption ensured that data remained confidential and tamper-proof during transmission and storage. Security tests, including attempted interception and replay attacks, demonstrated the system's resilience. The use of TLS encryption for MQTT communication and secure key management practices added additional layers of security against network-based threats.

### F. Resource Utilization

The Node MCU's limited resources were sufficient to handle the RC4 encryption and MQTT communication without significant strain. The memory usage remained within acceptable limits, and the power consumption was consistent with the device's specifications. The Raspberry Pi broker and the cloud function operated efficiently, with low CPU and memory usage during peak data transmission periods.

Power consumption measurements revealed that the encryption process drew approximately 0.5 W per message transmission, aligning with the Node MCU's operational specifications.

TABLE I  
PERFORMANCE METRICS OF THE ENCRYPTION FRAMEWORK

Metric	Sensor Node	Broker	Cloud Function
Encryption Time (ms)	5	4	N/A
Decryption Time (ms)	N/A	4	15
Memory Usage (KB)	20	50	N/A
Power Consumption (W)	0.5	2	N/A

### G. Encryption and Decryption Speed

The encryption and decryption speeds of RC4 at the sensor node, MQTT broker, and cloud function levels were measured, with results showing an average encryption time of 5 ms per message at the sensor node and 4 ms at the broker. The cloud function decrypted data in approximately 15 ms per request. This rapid processing enabled real-time data handling, crucial for responsive agricultural applications. The low-latency nature of RC4 and the efficient execution of the cloud function proved advantageous, enabling efficient, secure data flow without disrupting the timely transmission required for irrigation control.

### H. Memory and Power Consumption

The memory footprint of RC4 was found to be minimal, using less than 20 KB of memory on average per IoT node, well within the limits of low-power microcontrollers. Power consumption measurements revealed that the encryption process drew approximately 0.5 W per message transmission, a manageable level that supports prolonged operation in battery-powered IoT nodes. These findings align with the objectives of lightweight cryptography, confirming RC4 as an optimal choice for applications where both memory and power are limited resources.

### *I. Data Integrity and Security Resilience*

The multi-stage encryption and decryption process—encryption at the sensor node, decryption and re-encryption at the MQTT broker, and decryption via the cloud function—provided robust data protection. Data integrity checks conducted at the cloud platform confirmed that sensor data remained accurate and unaltered throughout its transmission journey. Simulated interception attempts during transmission and storage showed that the layered encryption effectively protected data from unauthorized access.

### *J. Comparative Analysis with Other Lightweight Protocols*

Comparing RC4's performance with the Expeditious Cipher and the RC4-ECC-SHA256 hybrid model demonstrated RC4's advantage in low-resource applications. While the Expeditious Cipher offered comparable encryption efficiency, RC4 maintained a smaller memory footprint, which is preferable for constrained devices in agricultural settings. The RC4-ECC-SHA256 model, though providing stronger encryption due to layered algorithms, exhibited longer encryption/decryption times and higher memory usage, making it less suitable for real-time applications. This comparative analysis highlights RC4's advantage in balancing speed, memory efficiency, and security within the context of IoT-based agriculture.

Although RC4 is known to have vulnerabilities, in our application, the risks are mitigated by using unique session keys for each transmission and limiting the amount of data encrypted with the same key. Future work will explore replacing RC4 with more secure lightweight ciphers such as AES-128 in CTR mode or ChaCha20.

## IV. DISCUSSION

The results indicate that the RC4-based encryption framework, enhanced with a cloud function for decryption, is well-suited to the specific requirements of smart irrigation systems, where resource limitations and real-time response are critical. By leveraging RC4's speed and low memory usage, and utilizing cloud computing resources for secure decryption, the proposed framework secures data transmission while maintaining the responsiveness needed for effective irrigation control. The multi-stage encryption process addresses common IoT security threats, such as eavesdropping and man-in-the-middle attacks, without imposing significant computational burdens on the devices involved.

One limitation of this study is the scalability of the MQTT broker on the Raspberry Pi when handling a large number of sensor nodes. Future research will focus on load balancing techniques and distributed broker architectures to enhance scalability. Additionally, while the cloud function adds a layer of security, it introduces dependency on cloud service availability and potential latency, which will be explored in further studies.

The successful implementation of this RC4 framework with cloud-based decryption points to broader applications in

precision agriculture and other IoT-driven fields requiring secure, low-power data handling, similar to secure architectures employed in smart cities [17]. Future research may explore combining RC4 with additional lightweight security protocols for enhanced resilience against evolving cyber threats. For larger IoT networks or systems with more extensive data security needs, hybrid encryption models incorporating RC4 and other lightweight ciphers could offer an effective solution.

In conclusion, this study confirms that lightweight cryptography, particularly RC4, combined with cloud-based decryption functions, provides a viable pathway for securing IoT applications in agriculture. The findings suggest that secure, efficient data handling in resource-constrained environments is achievable, supporting further advancements in precision agriculture and IoT-based data security frameworks.

## V. CONCLUSION

This study presents a secure and efficient data transmission framework for IoT-based smart irrigation systems, leveraging the RC4 encryption algorithm in conjunction with the MQTT protocol, Raspberry Pi as the broker, Node MCU microcontrollers as clients, and integration with Google Cloud Firestore, a cloud function for decryption, and Looker Studio. The implementation demonstrates that lightweight cryptography and communication protocols can effectively address the security and efficiency challenges inherent in precision agriculture.

The MQTT protocol facilitated reliable and low-latency communication between sensor nodes and the cloud [5], [11]. The Raspberry Pi broker efficiently managed message routing and provided a platform for additional security processing without introducing significant delays. The Node MCU clients, despite their resource constraints, successfully performed encryption and communication tasks, validating the suitability of RC4 and MQTT for such devices.

Integration with Google Cloud Firestore and the use of a cloud function for decryption ensured scalable and secure real-time data storage and retrieval [14], [18], while Looker Studio provided powerful visualization tools for monitoring and analysis. The system maintained data integrity and confidentiality throughout the data flow, from sensor nodes to cloud visualization, supporting informed decision-making in irrigation management.

The successful deployment of this framework highlights the potential for combining lightweight cryptography with efficient communication protocols and cloud services to enhance security in IoT-based agricultural applications. Future work may explore further optimization of encryption algorithms, incorporation of machine learning for predictive analytics, and expansion to other areas of precision agriculture.

The implementation confirms that lightweight cryptography combined with MQTT protocol and cloud services can effectively secure data transmission in IoT-based smart irrigation systems without imposing significant resource demands.

## REFERENCES

- [1] C. Fathy and H. M. Ali, "A Secure IoT-Based Irrigation System for Precision Agriculture Using the Expeditious Cipher," *Sensors*, vol. 23, no. 2091, pp. 1–16, Feb. 2023. DOI: 10.3390/s23042091.
- [2] S. K. Mousavi, A. Ghaffari, S. Besharat, and H. Afshari, "Security of Internet of Things Using RC4 and ECC Algorithms (Case Study: Smart Irrigation Systems)," *Wireless Personal Communications*, vol. 116, no. 3, pp. 1713–1742, Sep. 2021. DOI: 10.1007/s11277-020-07758-5.
- [3] United Nations Food and Agriculture Organization (FAO), "The Future of Food and Agriculture: Trends and Challenges," FAO Publications, 2017.
- [4] International Organization for Standardization (ISO), "ISO/IEC 29192: Information Technology—Security Techniques—Lightweight Cryptography," ISO Standards, 2012.
- [5] Message Queuing Telemetry Transport (MQTT), "ISO/IEC 20922: Information Technology—Message Queuing Telemetry Transport (MQTT) Protocol," ISO Standards, 2016.
- [6] F. Tschorsch and B. Scheuermann, "Bitcoin and Beyond: A Technical Survey on Decentralized Digital Currencies," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 3, pp. 2084–2123, 2015.
- [7] J. Singh, T. Pasquier, J. Bacon, H. Ko, and D. Eysers, "Twenty Security Considerations for Cloud-Supported Internet of Things," *IEEE Internet of Things Journal*, vol. 3, no. 3, pp. 269–284, 2016.
- [8] E. Borgia, "The Internet of Things vision: Key features, applications, and open issues," *Computer Communications*, vol. 54, pp. 1–31, 2014. DOI: 10.1016/j.comcom.2014.09.008.
- [9] K. Fan, Y. Gong, H. Li, and Y. Yang, "Lightweight and Secure ECC-Based RFID Authentication Scheme for Internet of Things," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 8, pp. 3759–3768, 2018. DOI: 10.1109/TII.2017.2773644.
- [10] W. Jiang, C. Zhang, and X. Xie, "Research on Security Technology of Internet of Things Based on Lightweight Encryption Algorithm," *Journal of Physics: Conference Series*, vol. 1570, no. 1, 2020. DOI: 10.1088/1742-6596/1570/1/012044.
- [11] N. Naik, "Choice of Effective Messaging Protocols for IoT Systems: MQTT, CoAP, AMQP and HTTP," *2017 IEEE International Systems Engineering Symposium (ISSE)*, Vienna, 2017, pp. 1–7. DOI: 10.1109/SysEng.2017.8088251.
- [12] E. Fernandes, A. A. Pereira, and L. A. Villas, "A MQTT Dynamic Reconfiguration Architecture for IoT Devices," *2018 14th IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, Limassol, 2018, pp. 1–8. DOI: 10.1109/WiMOB.2018.8589125.
- [13] S. D. T. Kelly, N. K. Suryadevara, and S. C. Mukhopadhyay, "Towards the Implementation of IoT for Environmental Condition Monitoring in Homes," *IEEE Sensors Journal*, vol. 13, no. 10, pp. 3846–3853, 2013. DOI: 10.1109/JSEN.2013.2263379.
- [14] Google Cloud, "Cloud Firestore Documentation," Available: <https://cloud.google.com/firestore/docs>
- [15] Google, "Looker Studio Documentation," Available: <https://cloud.google.com/looker/docs>
- [16] H. S. Kim, J. Lee, and S. Kim, "An Internet of Things (IoT) Application with MQTT Protocol," *2018 International Conference on Information Networking (ICOIN)*, Chiang Mai, 2018, pp. 714–717. DOI: 10.1109/ICOIN.2018.8343233.
- [17] E. S. Al-Shaikh and M. A. Al-Qutayri, "A Secure IoT Architecture for Smart Cities," *2018 International Conference on Computer and Applications (ICCA)*, Beirut, 2018, pp. 152–156. DOI: 10.1109/COMAPP.2018.8460350.
- [18] Google Cloud, "Cloud Functions Documentation," Available: <https://cloud.google.com/functions/docs>
- [19] Google Cloud, "Cloud Secret Manager Documentation," Available: <https://cloud.google.com/secret-manager/docs>