sensors

MDPI

Article

A Secure IoT-Based Irrigation System for Precision Agriculture Using the Expeditious Cipher

Cherine Fathy [1,*] and Hassan M. Ali [2]

[1] Computer Engineering Department, College of Engineering and Technology, Arab Academy for Science and Technology (AAST), Alexandria 1029, Egypt
[2] National Training Academy, Sheikh Zayed City 12582, Egypt
* Correspondence: cherine_fathy@aast.edu

Abstract: Due to the recent advances in the domain of smart agriculture as a result of integrating traditional agriculture and the latest information technologies including the Internet of Things (IoT), cloud computing, and artificial intelligence (AI), there is an urgent need to address the information security-related issues and challenges in this field. In this article, we propose the integration of lightweight cryptography techniques into the IoT ecosystem for smart agriculture to meet the requirements of resource-constrained IoT devices. Moreover, we investigate the adoption of a lightweight encryption protocol, namely, the Expeditious Cipher (X-cipher), to create a secure channel between the sensing layer and the broker in the Message Queue Telemetry Transport (MQTT) protocol as well as a secure channel between the broker and its subscribers. Our case study focuses on smart irrigation systems, and the MQTT protocol is deployed as the application messaging protocol in these systems. Smart irrigation strives to decrease the misuse of natural resources by enhancing the efficiency of agricultural irrigation. This secure channel is utilized to eliminate the main security threat in precision agriculture by protecting sensors' published data from eavesdropping and theft, as well as from unauthorized changes to sensitive data that can negatively impact crops' development. In addition, the secure channel protects the irrigation decisions made by the data analytics (DA) entity regarding the irrigation time and the quantity of water that is returned to actuators from any alteration. Performance evaluation of our chosen lightweight encryption protocol revealed an improvement in terms of power consumption, execution time, and required memory usage when compared with the Advanced Encryption Standard (AES). Moreover, the selected lightweight encryption protocol outperforms the PRESENT lightweight encryption protocol in terms of throughput and memory usage.

Keywords: precision agriculture; lightweight encryption; MQTT; smart irrigation; IoT; expeditious cipher; AES; PRESENT; lightweight cryptography

Citation: Fathy C.; Ali, H.M. A Secure IoT-Based Irrigation System for Precision Agriculture Using the Expeditious Cipher. *Sensors* **2023**, *23*, 2091. https://doi.org/10.3390/s23042091

Academic Editors: Dimitrios Piromalis, Konstantinos G. Arvanitis and Panagiotis Papageorgas

Received: 5 January 2023
Revised: 3 February 2023
Accepted: 10 February 2023
Published: 13 February 2023

1. Introduction

An increasing number of research studies on smart agriculture has been motivated by several challenges. First, the explosive growth of the world population [1]. According to the United Nations (UN) Food and Agriculture Organization, an increase of up to 70% more food will be required in 2050. Second, the declining agricultural lands and exhaustion of finite natural resources such as fresh water and arable land. In addition, the decreasing number of agricultural laborers in the majority of countries. As a consequence of this agricultural workforce decline, there is an urgent need for the adoption of IoT solutions in agriculture practices to reduce the need for manual labor. IoT solutions support farmers to tighten the supply-demand gap.

Precision agriculture integrates wireless sensor networks (WSNs) with traditional agriculture to improve crop yields. This is carried out by using a large number of low-power multi-function wireless communication sensors to remotely monitor the farmland

to collect environmental data, crop growth data, and livestock health data to guarantee a reduction in the possible threats to the production process and help farmers make better decisions. Recently, there is a shift from the usage of WSNs for smart agriculture to the IoT as the main enabling technology for smart agriculture. The IoT combines several technologies such as radio frequency identification, wireless sensor networks, middleware systems, end-user applications, and cloud computing.

In [2,3], the authors presented an IoT ecosystem architecture for smart agriculture that is made up of four main components, namely: IoT devices, communication technology, internet, and data storage and processing. First, the IoT devices are responsible for monitoring the farming environment and collecting environmental data, crop growth data, and livestock health data. Second, the role of communication technology is to establish robust, reliable, and secure communication between the cloud and the farms. Wireless communication standards are categorized based on the coverage range into short-range and long-range standards. The short-range standards include Bluetooth, near-field communication (NFC)-enabled devices, ZigBee, Z-Wave, and passive and active radio frequency identification (RFID) systems. The long-range communication standards are defined as low-power wide-area networks (LPWA). Examples of long-range communication standards include LoRa, Sigfox, and NB-IoT. The LPWA technologies provide a wide area of coverage to low-power devices [4]. LPWA technologies outperform conventional cellular and short-range wireless technologies for different emerging smart city and machine-to-machine applications such as metering, logistics, industrial monitoring, and agriculture. However, LPWA technologies realize long-range ranging from a few to tens of kilometers and low-power operations at the expense of a low data rate. The primary aim of LPWA technologies is to achieve a 10-year battery life. LPWA technologies are suitable for delay-tolerant applications, as they achieve throughput in orders of ten kilobits per second and high latency in orders of seconds or minutes. Long range is achieved due to the use of the sub-1GHz band, and the deployment of narrow-band and spread spectrum techniques are the modulation techniques adopted by different LPWA technologies. In [4], the authors compared the technical specifications of various LPWA technologies and standards. The choice of communication technology depended on the application of the IoT device and the type of topology. Third, the internet is the core network layer enabling the availability of data collected by IoT devices anywhere and anytime. Routing Protocol for Low Power and Lossy Networks (RPL) [5] has been standardized by the Internet Engineering Task Force (IETF) as a routing protocol for resource-constrained nodes in IoT. RPL builds a robust topology over lossy links. A destination-oriented directed acyclic graph (DODAG) is the core of RPL, which represents a routing diagram of nodes. In [6], the authors proposed an enhanced routing protocol based on RPL called E-RPL that decreases the number of control messages. Moreover, they proposed a flexible multi-constrained objective function (OF) that integrates several metrics such as energy, delay, and bandwidth to define the end-to-end path between the sink and a given node. The simulation results of their proposal revealed a remarkable improvement in terms of end-to-end delay, energy consumption, and routing overhead. Finally, different platforms have been developed to provide data analytics, data management, and data storage of the big data collected from sensors. Data analytics (DA) has a primary role in improving the efficiency of smart agriculture systems and in increasing productivity. DA is classified into five classes based on the requirements of IoT applications: real-time analytics, memory-level analytics, offline analytics, business intelligence-level analytics, and massive analytics. In [7], the authors presented big IoT data analytic types, methods, and technologies for big data mining. DA can help in insurance, prediction, storage management, decision-making, farm management, and precision farming. In irrigation systems, the automated decision made by DA controls the water supply timing and quantity. The main objective of big data analytics is to analyze collected information to predict and identify recent trends, find hidden information, and finally, make decisions. Prediction, classification, clustering, and association rules are the main big data analytics methods.

Furthermore, smart irrigation is a precision agriculture application [8] that aims to control water consumption in the agriculture sector as a scarce resource in many countries by deploying IoT technology to remotely gather information from sensors implanted in agriculture terrains to monitor soil different parameters in all stages. Based on the collected information, a decision is made on when to irrigate and the water quantity and quality required. Table 1 reviews the recent research on IoT-based precision irrigation systems, highlighting the communication protocol, data analytics, and security techniques deployed in these systems. As can be noticed, recent research ignores security techniques and concentrates on the usage of machine learning and artificial intelligence approaches, such as fuzzy logic, artificial neural networks (ANNs), and regression models, to optimize IoT-based irrigation systems employing different environmental parameters and weather conditions to schedule the irrigation timing and the quantity of water used for irrigation. However, any alteration in the information coming from sensors and decisions passed to actuators can lead to crop damage, which is considered to be a crucial threat to the national security of any country. As such, a secure channel must be developed between the sensing layer and the decision-making entity to secure information flowing from the sensors to the decision-maker entity and to secure the decision returned to the actuators that control the irrigation system. Because precision agriculture is highly dependent on data and information from the monitored system, any alteration in such data during runtime can lead to expensive unmanageable decisions and actions from farmers. Therefore, there is a need to adopt the security mechanisms required to guarantee basic security functions: authenticity, reliability, integrity, and availability. Moreover, these security mechanisms must be lightweight to meet the requirements of constrained devices used in IoT.

Table 1. Literature review of recently proposed IoT-based precision irrigation systems.

Proposed Model	Security Techniques Used?	Communication Technology Deployed	Data Analytics (DA) Techniques Used?	Challenge
Machine Learning-based Irrigation System (2019) [9]	Not used	LoRa P2P	Multiple linear regression algorithm used to calculate the amount of water required for each irrigation	Lack of security techniques
The AREThOU5A IoT Platform (2021) [10]	Yes, TCP/IP SSL for encryption	LoRaWAN	Not used	The research lacks evaluation of the proposed security techniques and does not apply lightweight security algorithms as TLS techniques add overhead in terms of memory and energy on constrained nodes
Deep Learning Intelligent Irrigation System for Agriculture (DLISA) (2021) [11]	Not used	Wireless communication but not specified	LSTM RNN model is proposed to predict the volumetric soil moisture of the next day to schedule irrigation	Lack of security techniques
Sensor-based Smart Irrigation System (2021) [12]	Not used	WiFi	ThingsSpeak displays graphs of collected soil parameters, irrigation decision is made by farmers to activate/deactivate the irrigation	Lack of security techniques
Secure Multi-Crop Smart Irrigation System (SMCSIS) (2021) [13]	Yes, access control techniques & Blockchain technology for privacy and data integrity	Not specified	A supervised feed-forward neural network (FFNN) to estimate evaporation for estimated soil moisture (ESM) calculation to specify the next irrigation time.	The research lacks evaluation of the proposed security techniques
Machine learning-based Irrigation System (2022) [14]	Not used	Lora	Support vector machine (SVM) and k-nearest neighbor (KNN) algorithms used for prediction of irrigation scheduling	Lack of security techniques

Table 1. *Cont.*

Proposed Model	Security Techniques Used?	Communication Technology Deployed	Data Analytics (DA) Techniques Used?	Challenge
Intelligent Sensor-based Smart Irrigation System (2023) [15]	Not used	Lora	Recurrent neural network to forecast the soil moisture level	Lack of security techniques
Our proposed model Secure IoT-based Irrigation System (2023)	Yes, The Expeditious Cipher (X-cipher)	WiFi and LoRAWAN or GSM	ThingsSpeak displays graphs of collected soil parameters	The proposed model integrates lightweight cryptography to the IoT ecosystem of smart irrigation systems

In [1], the authors reviewed the previous work conducted on smart agriculture and highlights different aspects of applying IoT solutions in smart agriculture. Moreover, the article reviews smart agriculture's related security issues and compares security issues in the industry (urban) and agriculture (rural).

In [16], the authors presented a classification of security threats in smart agriculture and precision agriculture environments. The authors classified security threats into six possible attacks: attacks on hardware (side channel attack and radio frequency (RF) jamming), attacks on the network equipment (denial of service (DoS), MITM (man in the middle), botnets, cloud computing attacks), attacks on data (data leakage, ransomware, cloud data leakage, false data injection, misconfiguration), attacks on applications (software update attacks, malware injection, buffer overflow, indirect attacks (SQL injection)), attacks on support chain (third-party attacks, data fabrication), and misuse attacks (cyber-terrorism, invalidation, and compliance). Another classification of security threats is underlined in [17]. The authors classified the security requirements in smart agriculture into six challenges, namely: integrity, availability, authentication, confidentiality, privacy, and Non-repudiation, and highlighted the possible attacks under each challenge. Moreover, a review of existing solutions to IoT security problems is emphasized. In [8], the authors added data freshness, authorization, and self-healing to the security requirements of smart agriculture. In [18], the authors reviewed IoT communication technologies security aspects for smart agriculture. In [19], the authors reviewed all categories of security attacks and the application of WSNs in IoT along with an evaluation of the countermeasures adopted against each type of attack.

Moreover, smart irrigation systems developed based on IoT technology consist of constrained nodes in terms of power, memory, and processing resources. As a result, conventional security protocols can not be supported in such systems [20,21]. Transport layer security (TLS) adds overhead in terms of memory and energy on constrained nodes. As a result of the dependence on constrained nodes in IoT-based irrigation systems, a lightweight security protocol must be deployed. Lightweight cryptography techniques balance throughput against power drain, memory usage, and gate equivalent and have lower performance when compared to cryptography standards (such as AES and SHA-256) [22]. Characteristics of lightweight cryptography are highlighted in ISO/IEC 29192 and ISO/IEC JTC 1/SC 27. Lightweight properties are evaluated based on chip size and energy consumption and small code and/or RAM size in case of software implementation [21]. In [21], the authors discussed privacy in IoT in the context of developing solutions and frameworks that address profiling and tracking, localization, and tracking challenges and underlined state-of-the-art lightweight cryptographic framework for IoT. In [23], the authors proposed a set of lightweight security protocols for encryption, authentication, and key management for IoT. The authors compared their proposed protocols with IPsec in terms of security and computational efficiency. They succeeded in achieving a decreased level of resource consumption with an increased level of security. In [24], the authors implemented AES and PRESENT ciphers on a smartphone and provided a performance evaluation comparison between the two algorithms. AES is the symmetric block cipher defined by the National Institute of Standards and Technology (NIST) as the standard for bulk data encryption,

whereas PRESENT is a symmetric ultra-lightweight block cipher that was standardized by ISO/IEC.

On the other hand, the Message Queue Telemetry Transport (MQTT) protocol has been widely deployed as an application layer messaging and information exchange protocol in machine-to-machine (M2M) communication [25]. This is due to its ability to function with resource-constrained devices that utilize low bandwidth and unreliable links. MQTT is considered a lightweight, energy-efficient, and bandwidth-efficient communication protocol. MQTT utilizes the publish/subscribe architecture model to provide transition flexibility and simplicity of implementation. MQTT's main components are the publishers (lightweight sensors), the subscribers (applications interested in sensor data), and the brokers (connect publishers and subscribers and classify sensor data into topics) as illustrated in Figure 1. The data generated by a publisher are dispatched to multiple subscribers through an MQTT broker. MQTT was proposed in 1999 by Andy Stanford-Clark of IBM and Arlen Nipper of Arcom and is currently an OASIS (Organization for the Advancement of Structured Information Standards) standard; it also has a standard defined in ISO/IEC 20922:2016.

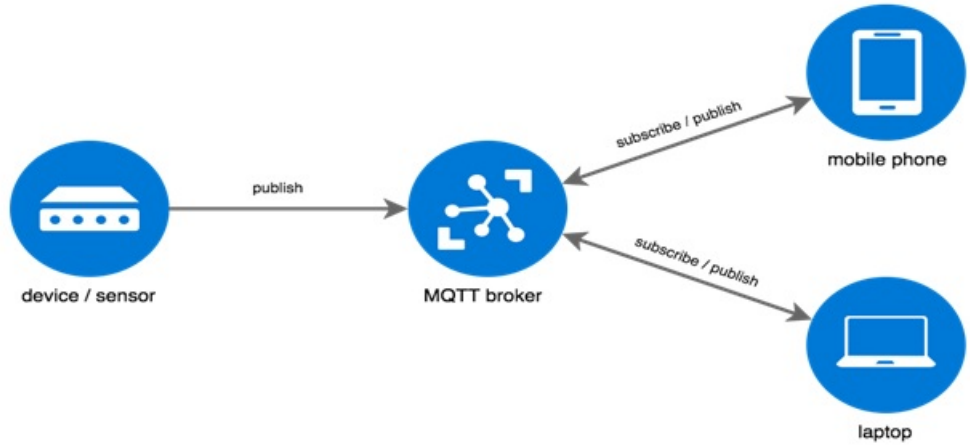

Figure 1. MQTT architecture.

Motivated by the increasing importance of smart irrigation systems in conserving water as a scarce natural resource, the role of precision agriculture in agriculture development, and the urgent need to apply information security techniques in the IoT part of the smart agriculture ecosystem, our aim in this research is to integrate a lightweight cryptography layer into the IoT ecosystem for smart agriculture and to investigate the deployment of a lightweight encryption protocol (the Expeditious Cipher) to create a secure channel between the sensing layer and the broker of MQTT protocol as well as between the broker and its subscribers in smart irrigation systems. This secure channel protects the sensors' published sensitive data from eavesdropping and theft and preserves the integrity of data, in addition to protection of the decision that is made by the DA entity and returned to actuators. It should be noted that the security in IoT-based systems lies in IoT local systems consisting of devices constrained in energy and computing power.

The following points summarize the main contributions of this article.

1. The main contribution of this article is the integration of a lightweight cryptography layer to the IoT ecosystem for smart agriculture that meets the requirements of constrained devices used in smart agriculture in general and specifically, in our proposed IoT-based irrigation system.
2. The article investigates the deployment of a lightweight encryption protocol (Expeditious Cipher (X-cipher)) to create a secure channel between the sensing layer and the broker in the MQTT protocol as well as a secure channel between the broker and its subscribers in smart irrigation systems (our case study).
3. The proposed model is evaluated through simulation to validate the lightweight property of the chosen encryption protocol in terms of power consumption, execu-

tion time, and memory usage. Moreover, a performance comparison is carried out between the Expeditious Cipher (X-cipher), AES, and PRESENT cipher (lightweight standard protocol) in terms of power consumption, execution time, memory usage, and average throughput.

4. The security requirements of IoT-based agriculture systems and the potential attacks against them are discussed.
5. A state-of-the-art lightweight security architectures proposed for securing MQTT protocol is reviewed after highlighting the concept of lightweight cryptography.

The remainder of the article is organized as follows: Section 2 presents our proposed secure smart irrigation system after briefly reviewing the state-of-the-art proposed lightweight security architectures for securing MQTT protocol. Section 3 highlights and discusses the performance evaluation results of our selected lightweight encryption algorithm versus AES, in addition to a performance comparison between X-cipher and the PRESENT cipher. Section 4 summarizes the findings of the article and highlights our suggestions for future work.

2. Materials and Methods

In this section, the proposed secure IoT-based irrigation system architecture is outlined. First, a brief review of the proposed lightweight security techniques for securing the MQTT protocol is given. Then, the proposed model architecture and hardware components are explained in detail. In addition, the applicability of the selected lightweight protocol (X-Cipher) on NodeMCU is verified.

2.1. MQTT Security Challenges and Solutions

MQTT runs over TCP with few security techniques such as simple authorization policies and basic authentication techniques. TLS/SSL with session key management has been suggested for securing MQTT; however, proposed security techniques for MQTT should be lightweight for deployment in IoT environments. TLS/SSL has been proven to have performance issues in terms of processing time, memory required, and energy consumption, in addition to the various possible attacks such as Heartbleed, CRIME, BEAST, and RC4. Furthermore, TLS does not provide fine-grained access control [26]. Moreover, MQTT suffers from several security shortcomings, mainly from a lack of confidentiality, integrity, availability (DOS attack), mutual authentication, access control, control message security, and end-to-end security. In [27], the authors reviewed the symmetric, asymmetric, and hybrid lightweight schemes proposed in the literature for guaranteeing the confidentiality of transmitted data using MQTT. Moreover, the authors' review suggested security techniques for guaranteeing access control in MQTT. According to [28], the security threats against MQTT are mainly replay attacks and man-in-the-middle attacks (MITM). Value-to-keyed-hash message authentication code (Value-to-HMAC) has been deployed to achieve information confidentiality. In [26], the authors designed a security architecture for MQTT-SN to achieve end-to-end security in addition to fine-grained access control. Moreover, they introduced a certificate subject to create a secure direct channel between a publisher and a set of subscribers. In addition, they designed security schemes to integrate mutual authentication and control message security functions into the standard MQTT-SN control messages, without relying on the TLS. In [20], the authors proposed ChaCha20-Poly1305 Authenticated Encryption with Associated Data (AEAD) as a solution to secure resource-constrained node communication over MQTT/MQTT-SN. RFC 6574 [29] describes constrained nodes as they are bandwidth constrained, energy constrained, and memory constrained. Performance evaluation of the proposed scheme revealed a low memory requirement and low processing time. In [30], the authors presented a lightweight authentication and encryption mechanism based on ECDHE-PSK (Elliptic Curve Diffie–Hellman Ephemeral)-(Pre-Shared Key) for IoT-constrained devices. The proposed security mechanism outperforms the ECDHE-ECDSA in all performance evaluation tests.

2.2. Proposed IoT System Architecture

The main objective of our proposed system is the addition of a lightweight cryptography layer to the agriculture IoT ecosystem, as shown in Figure 2. The function of this layer is to create a secure communication channel between the sensing/actuator layer (IoT nodes) and the subscriber layer, thus protecting sensors' published data from eavesdropping and theft, as well as from unauthorized changes to sensitive data that can negatively impact crop development. In addition, the secure channel protects the irrigation decision made by the data analytics (DA) entity regarding the irrigation time and the quantity of water that is returned to actuators from any alteration while meeting the requirements of IoT-constrained devices in terms of memory usage and power consumption by ensuring the application of lightweight cryptography techniques.

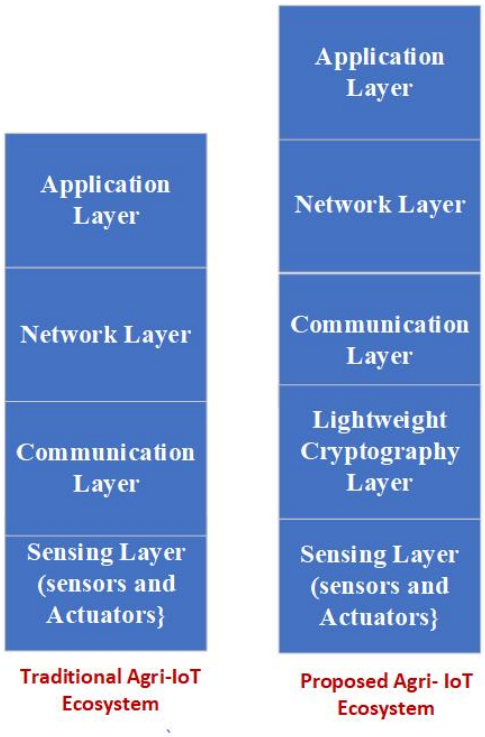

Figure 2. Traditional Agri-IoT ecosystem versus proposed Agri-IoT ecosystem.

Figure 3 depicts the architecture of our proposed secure IoT-based smart irrigation system. Figure 3a illustrates the general system architecture and Figure 3b shows the hardware components of the proposed system. The proposed architecture consists of five layers: the sensing/actuator layer, the lightweight cryptography layer, the communication layer, the network layer, and the application layer.

2.2.1. The Sensing Layer

The sensing layer is made up of multiple low-power sensors that are spread on agricultural land to continuously collect soil temperature and humidity. The DHT11 sensors are used to measure the temperature value in degrees Celsius and the humidity value in percentage. The DHT-11 sensor has three pins: two of which are for power and one is for output data transmission. The three pins are used to connect the DHT11 sensor to the NodeMCU ESP8266 board. The DHT-11 library must be added to the Arduino IDE software before uploading to the NodeMCU ESP8266 board the code written to read the humidity and temperature value from the DHT11 sensor. Together, the DHT-11 sensor and the NodeMCU form the IoT node. The NodeMCU runs the MQTT client code and publishes the sensed soil parameters (the temperature and humidity readings) to the MQTT broker after encrypting the message using the Expeditious Cipher (X-cipher). The MQTT broker runs on Raspberry Pi, which is a low-power single-board computer. NodeMCU

ESP8266 has a built-in WiFi module that is used as the communication protocol between NodeMCU and the Raspberry Pi. The MQTT protocol is chosen due to its short message transmission capability and low-bandwidth requirement, which makes it more convenient for machine-to-machine (M2M) communication. Moreover, the Raspberry Pi was selected as a processing layer because of its low cost and common use in IoT applications. Mosquitto is an open-source lightweight MQTT broker and is selected as it is suitable for Raspberry Pi. The Mosquitto broker decrypts the message and adds the information in the message to the related topic. Then, information is encrypted again before being published (broadcast) to all subscribers. The ThingSpeak cloud platform is used among subscribers that execute cloud analysis for making irrigation decisions regarding the time of irrigation and the quantity of water. Then the irrigation decision is published to the broker to be decrypted and then encrypted again before being published to the actuator. The nodeMCU at the actuator layer decrypts the message and then passes the control signal to the attached pump to activate the irrigation.

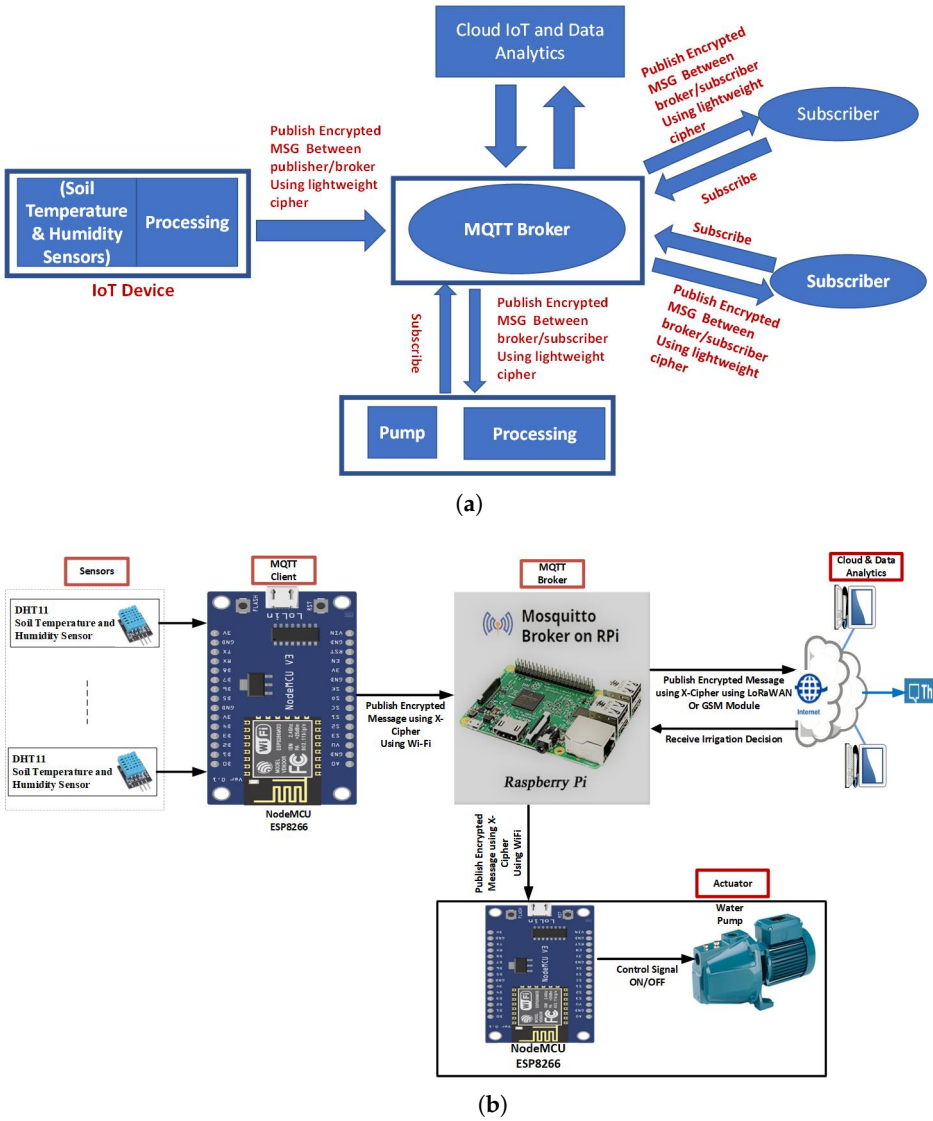

Figure 3. The proposed model. (**a**) proposed model architecture. (**b**) proposed model hardware.

2.2.2. Lightweight Cryptography Layer

In this layer, a lightweight protocol must be deployed, because the proposed IoT-based irrigation system is made up of constrained devices in terms of memory and energy. Characteristics of lightweight cryptography were highlighted in ISO/IEC 29192 and ISO/IEC

JTC 1/SC 27. Lightweight properties are evaluated based on chip size and energy consumption, and small code and/or RAM size in case of software implementation [21]. So, the Expeditious Cipher (X-cipher) was selected [31]. The X-cipher was proposed in 2011 and is considered a lightweight high-throughput encryption protocol. The cipher sub-keys are generated using the well-studied SHA-512 and Whirlpool 512 hash functions. The two hash functions are cascaded in a pseudo-random manner that depends on the user key to enhance the cipher security. The cipher utilizes a variable-size user key. The encryption rate can attain 512 bits per cycle in the case of hardware implementation with parallelization encryption paths. Eight rounds are recommended for the proper operation of the Expeditious Cipher (X-cipher). The circuitry of the encryptor and decryptor are identical, as such, a high code density and small implementation area are achieved, as required by lightweight cryptography. An additional important feature of the X-cipher is the separation of key scheduling process from the encryption process, which allows the change of cipher design, key, and block size by simply changing the hash function. The Expeditious Cipher (X-cipher) complete encryption algorithm is presented in [31]. NodeMCU has a cryptography module called crypto that contains various functions for working with cryptographic algorithms. The AES 128-bit is supported in ECB and CBC modes, in addition to several hash functions, namely, MD5, SHA-1, SHA-256, SHA-384, and SHA-512. The MQTT module implemented in C language is available on the NodeMCU ESP8266 board to run the MQTT client to publish the temperature and humidity readings to the MQTT broker running on Raspberry Pi. So, because the proposed Expeditious Cipher (X-cipher) depends on the SHA-512 hash function that is already implemented in the crypto module and the modulo 2 addition between the hashed key and the plain text, the selected lightweight algorithm can be compiled easily on the NodeMCU board. Table 2 compares the operational features of AES (traditional encryption protocol), PRESENT (lightweight standard protocol), X-cipher (selected lightweight protocol), and the lightweight encryption protocol proposed recently in the literature.

Table 2. Comparison between X-cipher, AES, PRESENT, and lightweight encryption.

Protocol	Type	Key	Throughput	Number of Rounds	Comment
AES [32]	Symmetric block cipher	Different key for each round	128 bits per round	10	Security depends on bit permutation and S-box
X-cipher [31]	Symmetric block cipher	Different key for each round	512 bits per round	8	Security depends on cascaded hashed functions (SHA-512 and Whirlpool-512) to generate cipher sub- keys
PRESENT [33]	Symmetric block cipher	Different key for each round	64 bits per round	31	Security depends on bit permutation and S-box
Lightweight encryption [23]	Symmetric block cipher	Different key for each file	Not specified	Not specified	Security depends on secure key management protocol

2.2.3. Communication Layer

The communication between NodeMCU ESP8266 and the Raspberry Pi is achieved using WiFi, as the NodeMCU has a built-in WiFi module. On the other hand, the communication between the Raspberry Pi and the internet can be carried out using LoRaWAN technology taking into consideration the low data rate characteristic of LPWA technologies discussed in Section 1 or through the GSM module.

3. Results

In this section, we validate through simulation that the chosen lightweight cipher satisfies the requirements of constrained devices in IoT-based smart irrigation systems in terms of power consumption and memory usage. Moreover, we compare the chosen lightweight cipher with PRESENT, a standard lightweight protocol, and AES, a standard encryption protocol.

3.1. Simulation Scenario

Table 3 presents the simulation packages used in the performance evaluation of the X-cipher. A Windows 10 virtual machine runs the Raspberry Pi emulator that was selected. An Ubuntu 20.1 virtual machine is used for the subscriber. The DHT-11 sensors are connected directly to the Raspberry Pi Emulator. Figure 4 illustrates selected parts of the Python code used to obtain humidity and temperature reading from DHTT-11 sensors. The Adafruit_DHT Python library is available for reading the DHT series of humidity and temperature sensors on a Raspberry Pi. First, the sensor type and the pin that the DHT-11 is connected through to Raspberry Pi must be specified. Then, the function Adafruit_DHT.read_retry is used to read from the specified pin. As can be noticed, the humidity and temperature readings are encrypted using the lightweight class function encrypt, which is our class implementation for the X-cipher protocol. Moreover, Figure 5 depicts a selected part of the MQTT client code for the Raspberry Pi emulator. The paho.mqtt.client Python library is deployed to provide a client class with support for MQTT protocol, and it also provides some helper functions that facilitate publishing messages to an MQTT server. Figure 6 highlights selected parts of the python code used to calculate memory usage. Calculations of memory usage rely on the process.memory_info and choosing the "rss" option returns the actual physical memory the process is using. Figure 7 illustrated selected parts of the Python code used to calculate power consumption. The pyRAPL Python library is utilized to measure the energy footprint of a host machine along with the execution of a piece of Python code. Three performance metrics were used to evaluate and compare the Expeditious Cipher (X-cipher) and the AES cipher [32]: power consumption, execution time, and memory usage for a file of 10,000 message comma separated of total size 20 MB.

Table 3. Software versions deployed in performance evaluation.

Software	Description
Python 2.7.18	Programming language
Raspberry Pi emulator 4.19.0-13-amd64	Raspberry Pi emulator
Ubuntu 20.1	Virtual machine
Windows 10	Virtual machine
Mosquito	Open source MQTT broker
Raspberry Pi OS with desktop Debian version: 11 (bullseye)	Raspberry Pi operating systems

```
import Adafruit_DHT
.
.
# Set sensor type : Options are DHT11,DHT22 or AM2302
sensor = Adafruit_DHT.DHT11

# Set GPIO sensor is connected to
gpio = 4
.
.
rc = 0
while rc == 0:
    rc = mqttc.loop()
    # Use read_retry method. This will retry up to 15 times to
    # get a sensor reading (waiting 2 seconds between each retry).
    humidity, temperature = Adafruit_DHT.read_retry(sensor, gpio)
    # Reading the DHT11 is very sensitive to timings and occasionally
    # the Pi might fail to get a valid reading. So check if readings are valid.
    if humidity is not None and temperature is not None:
        print('Temp={0:0.1f}*C  Humidity={1:0.1f}%'.format(temperature, humidity))
        humidity=light_weight.encrypt()
        temperature = light_weight.encrypt()
        mqttc.publish("humidity", str(humidity))
        mqttc.publish("Temp", str(temperature))
        time.sleep(1)
    else:
        print('Failed to get reading. Try again!')
```

Figure 4. The Raspberry Pi emulator code for obtaining humidity and temperature readings from sensors.

```
import paho.mqtt.client as paho
.
.
def on_connect(self, mosq, obj, rc):
    self.subscribe("Fan", 0)

def on_message(mosq, obj, msg):
    print(msg.topic + " " + str(msg.qos) + " " + str(msg.payload))
    # Give 5 second delay

def on_publish(mosq, obj, mid):
    print("mid: " + str(mid))

def on_subscribe(mosq, obj, mid, granted_qos):
    print("Subscribed: " + str(mid) + " " + str(granted_qos))


mqttc = paho.Client()  # object declaration
# Assign event callbacks
mqttc.on_message = on_message  # called as callback
mqttc.on_connect = on_connect
mqttc.on_publish = on_publish
mqttc.on_subscribe = on_subscribe
url_str = os.environ.get('MQTT_URL', 'tcp://192.168.10.5:1883')
url = urlparse.urlparse(url_str)
mqttc.connect(url.hostname, url.port)
```

Figure 5. MQTT client code for the Raspberry Pi emulator.

```
import os, sys
.
.

# inner psutil function
def process_memory():
      process = psutil.Process(os.getpid())
      mem_info = process.memory_info()
      return mem_info.rss

# decorator function
def profile(func):
      def wrapper(*args, **kwargs):

            mem_before = process_memory()
            result = func(*args, **kwargs)
            mem_after = process_memory()
            print("{}:consumed memory: {:,}".format(
                  func.__name__,
                  mem_before, mem_after, mem_after - mem_before))

            return result
      return wrapper

# instantiation of decorator function
@profile
```

Figure 6. Memory usage calculation function.

```
import meter as meter
import pyRAPL
pyRAPL.setup()
    meter = pyRAPL.Measurement('bar')
    meter.begin()
    csv_output = pyRAPL.outputs.CSVOutput('result.csv')

    @pyRAPL.measure(output=csv_output)
    def encrypt(text, key, csv_output=None):
      .
      .
        meter.end()
        csv_output.save()
        return ciphertext
    pyRAPL.setup()
    meter = pyRAPL.Measurement('bar')
    meter.begin()
    csv_output = pyRAPL.outputs.CSVOutput('result.csv')

    @pyRAPL.measure(output=csv_output)
    def decrypt(ciphertext, key, csv_output=None):
        .
      | .

        meter.end()
        csv_output.save()
```

Figure 7. Power consumption calculation function.

Figure 8 compares the two algorithms' power consumption during the encryption and decryption process. As shown in the graph, X-cipher requires less power consumption than AES throughout the whole encryption and decryption process. This makes X-cipher more convenient for IoT devices with limited energy resources. This is due to the dependency of X-cipher on just the XOR function between the generated hashed sub-key and the plain text, whereas AES depends on substitution–permutation operations. Moreover, the number of rounds in X-cipher is just eight, whereas the number of rounds is ten in the case of AES.

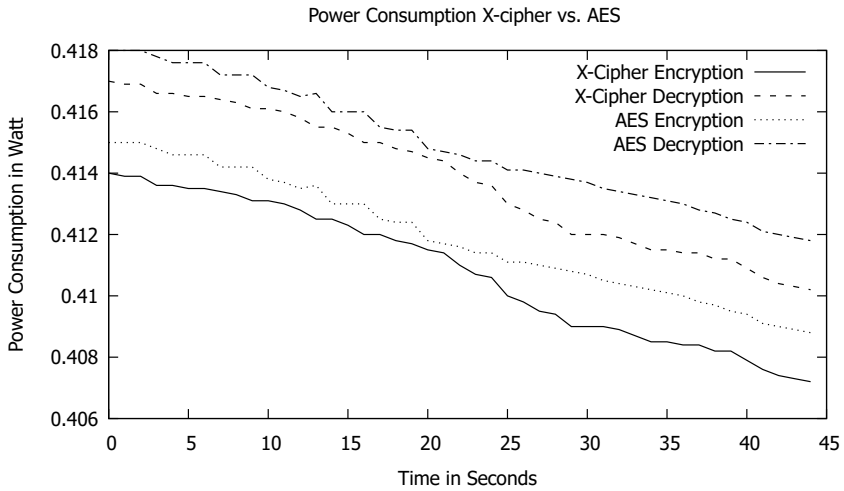

Figure 8. Power Consumption during encryption and decryption processes for AES and X-cipher.

Figure 9 compares the memory usage during the encryption and decryption process for the two algorithms. As shown in the graph, X-cipher requires less memory than AES throughout the whole encryption and decryption process. X-cipher has average memory usages of 20.04 MB and 22.38 MB during the encryption and decryption processes, respectively, in contrast to AES, which has average memory usages of 24.56 MB and 36.2 MB during the encryption and decryption processes, respectively. The higher memory usage in

the case of AES is due to the substitution–permutation principle employed in AES. This implies that X-cipher is more suitable for IoT devices with constrained memory capacity.

Figure 9 presents the average memory usage for both algorithms for the encryption and decryption of 20 MB messages. Figure 10 illustrates the average execution time for AES and X-cipher. As shown in both figures, the selected lightweight algorithm (X-cipher) outperforms AES in terms of average memory usage and average execution time.

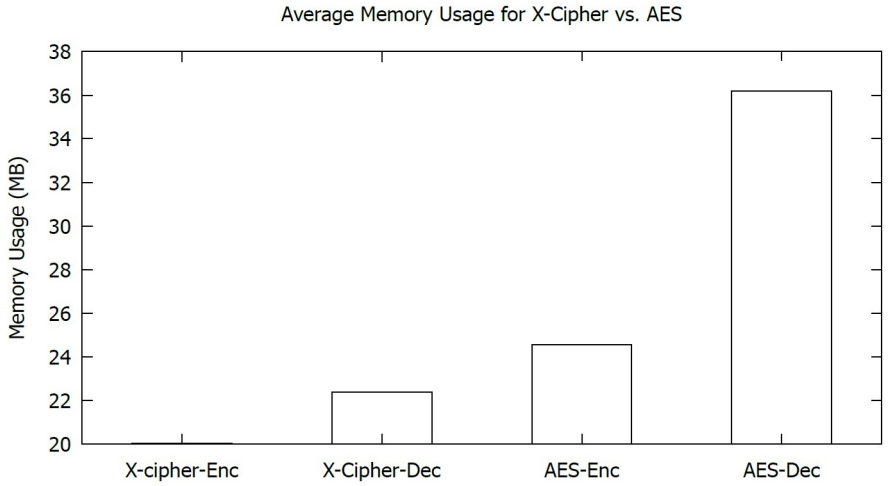

Figure 9. Average Memory usage for AES and X-cipher.

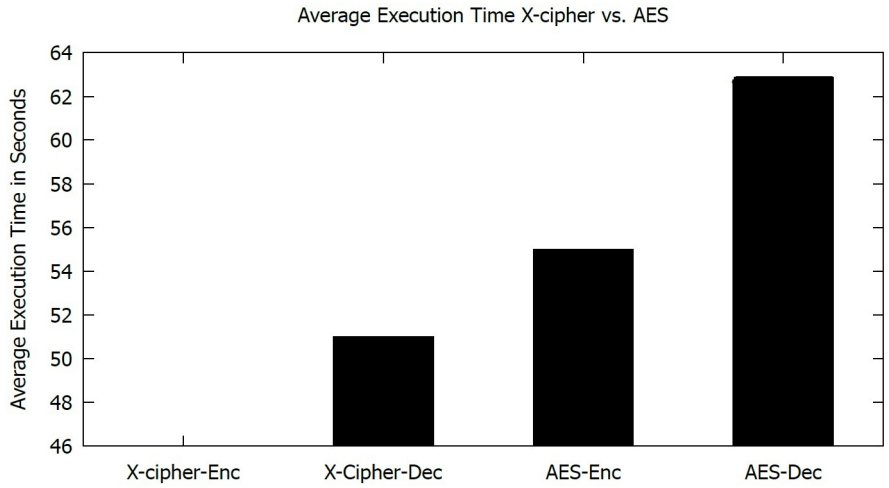

Figure 10. Average Execution time for AES and X-cipher.

3.2. Comparison with Related Work

In this section, we compare the performance of the X-cipher versus the PRESENT ultra-lightweight cipher [33] presented in the literature for the same size of data of 20 MB. Our selected algorithm X-cipher outperforms the PRESENT algorithm in terms of throughput and memory usage as shown in Tables 4 and 5. This is because PRESENT deploys the substitution permutation network principle although it has a reduced substitution box and due to the higher number of rounds, which is 31 rounds.

Table 4. Throughput results comparison between X-cipher and PRESENT.

Protocol-Process	Throughtput KBps
X-cipher-encryption	164 KBps
X-cipher-decryption	172 KBps
PRESENT-encryption	140 KBps
PRESENT-decryption	140 KBps
AES-encryption	200 KBps
AES-decryption	210 KBps

Table 5. Memory usage results in comparison between X-cipher and PRESENT.

Protocol-Process	Memory Usage (MB)
X-cipher-encryption	20.04 MB
X-cipher-decryption	22.38 MB
PRESENT-encryption	90 MB
PRESENT-decryption	105 MB

4. Conclusions

Smart irrigation systems integrate IoT technology with smart agriculture to conserve water consumption during the irrigation of agricultural land. This research focuses on the evaluation of the adoption of a lightweight security protocol to secure communication in smart irrigation systems. The adopted encryption algorithm (Expeditious Cipher) creates a secure channel between publishers and the broker of the MQTT protocol as well as between the broker and its subscribers. MQTT was chosen as the IoT application protocol for messaging in our proposed irrigation system, as it is a bandwidth- and energy-efficient protocol. The evaluated lightweight protocol reduces resource consumption in terms of memory usage and power consumption as well as minimizes the execution time, which makes it suitable for IoT-based resource-constrained systems. There is a trade-off between the security level and the memory and energy-processing resources required. AES achieves a higher security level at the cost of more memory usage and energy consumption, which causes performance degradation in constrained devices, opposite to lightweight cipher, which achieves an adequate level of security with low memory usage and reduced energy consumption. Comparing our selected X-cipher protocol with the PRESENT cipher reveals a lower memory usage and higher throughput in favor of the X-cipher. In the future, we will investigate the integration of our proposed system with software-defined networking (SDN) technology to implement the security models at the controller to offload publishers and brokers from the additional processing required by security algorithms. Moreover, we will develop a deep-learning model to predict the irrigation time and the required amount of water required for irrigation.

Author Contributions: Conceptualization, C.F. and H.M.A.; methodology, C.F. and H.M.A.; software, H.M.A.; validation, C.F. and H.M.A.; formal analysis, C.F. and H.M.A.; investigation, C.F. and H.M.A.; resources, C.F. and H.M.A.; writing—original draft preparation, C.F.; writing—review and editing, C.F.; visualization, C.F. and H.M.A.; project administration, C.F. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

Abbreviations

The following abbreviations are used in this manuscript:

AEAD	Authenticated Encryption with Associated Data
AES	Advanced Encryption System
AI	Artificial Intelligence
ANNs	Artificial Neural Networks
DA	Data Analytics
DOS	Denial Of Service
DODAG	Destination-Oriented Directed Acyclic Graph
DOS	Denial Of Service
ECDHE	Elliptic Curve Diffie Hellman Ephemeral
FFNN	Feed-Forward Neural Network
GSM	Global System for Mobile Communications
IoT	Internet of Things
IETF	Internet Engineering Task Force
PSK	Pre-Shared Key
LPWA	Low Power Wide Area
M2M	Machine to Machine
MITM	Man-In-The-Middle
MQTT	Message Queue Telemetry Transport
MQTT-SN	MQTT for Sensor Network
NFC	Near-Field Communications
NIST	National Institute of Standards and Technology
OASIS	Organization for the Advancement of Structured Information Standards
OF	Objective Function
RF	Radio Frequency
RFID	Radio Frequency Identification
RPL	Routing Protocol for Low-Power and Lossy Networks
SDN	Software-Defined Networking
SSL	Secure Socket Layer
TLS	Transport Layer Security
UN	United Nations
WSNs	Wireless Sensor Networks

References

1. Yang, X.; Shu, L.; Chen, J.; Ferrag, M.A.; Wu, J.; Nurellari, E.; Huang, K. A Survey on Smart Agriculture: Development Modes, Technologies, and Security and Privacy Challenges. *IEEE/CAA J. Autom. Sin.* **2021**, *8*, 273–302. [CrossRef]
2. Elijah, O.; Rahman, T.A.; Orikumhi, I.; Leow, C.Y.; Hindia, M.N. An Overview of Internet of Things (IoT) and Data Analytics in Agriculture: Benefits and Challenges. *IEEE Internet Things J.* **2018**, *5*, 3758–3773. [CrossRef]
3. Quy, V.K.; Hau, N.V.; Anh, D.V.; Quy, N.M.; Ban, N.T.; Lanza, S.; Randazzo, G.; Muzirafuti, A. IoT-Enabled Smart Agriculture: Architecture, Applications, and Challenges. *Appl. Sci.* **2022**, *12*, 3396. [CrossRef]
4. Raza, U.; Kulkarni, P.; Sooriyabandara, M. Low Power Wide Area Networks: An Overview. *IEEE Commun. Surv. Tutor.* **2017**, *19*, 855–873. [CrossRef]
5. Winter, T. ; Thubert, P.; Brandt, A.; Hui, J.; Kelsey, R.; Levis, P.; Pister, K.; Struik, R.; Vasseur, J.; Alexander, R. Rfc6550: Rpl. Internet Engineering Task Force (IETF), Request for Comments: 6550. 2012; pp. 1–157. Available online: https://www.rfc-editor.org/rfc/pdfrfc/rfc6550.txt.pdf (accessed on 30 December 2022).
6. Zier, A.; Abouaissa, A.; Lorenz, P. E-RPL: A Routing Protocol for IoT Networks. In Proceedings of the 2018 IEEE Global Communications Conference, GLOBECOM 2018—Proceedings, Abu Dhabi, United Arab Emirates, 9–13 December 2018. [CrossRef]
7. Marjani, M.; Nasaruddin, F.; Gani, A.; Karim, A.; Hashem, I.A.T.; Siddiqa, A.; Yaqoob, I. Big IoT Data Analytics: Architecture, Opportunities, and Open Research Challenges. *IEEE Access* **2017**, *5*, 5247–5261. [CrossRef]
8. Farooq, M.S.; Riaz, S.; Abid, A.; Abid, K.; Naeem, M.A. A Survey on the Role of IoT in Agriculture for the Implementation of Smart Farming. *IEEE Access* **2019**, *7*, 156237–156271. [CrossRef]
9. Chang, Y.C.; Huang, T.W.; Huang, N.F. A Machine Learning Based Smart Irrigation System with LoRa P2P Networks. In Proceedings of the 2019 20th Asia-Pacific Network Operations and Management Symposium: Management in a Cyber-Physical World, APNOMS 2019, Matsue, Japan, 18–20 September 2019; pp. 31–34. [CrossRef]
10. Boursianis, A.D.; Papadopoulou, M.S.; Gotsis, A.; Wan, S.; Sarigiannidis, P.; Nikolaidis, S.; Goudos, S.K. Smart Irrigation System for Precision Agriculture—The ARETHOU5A IoT Platform. *IEEE Sens. J.* **2021**, *21*, 17539–17547. [CrossRef]

11. Kashyap, P.K.; Kumar, S.; Jaiswal, A.; Prasad, M.; Gandomi, A.H. Towards Precision Agriculture: IoT-Enabled Intelligent Irrigation Systems Using Deep Learning Neural Network. *IEEE Sens. J.* **2021**, *21*, 17479–17491. [CrossRef]
12. Hassan, N.; Hasib Cheragee, S.; Ahammed, S.; Touhidul Islam, A.Z. Sensor based Smart Irrigation System with Monitoring and Controlling using Internet of Things. *Int. J. Ambient Syst. Appl.* **2021**, *9*, 17–26. [CrossRef]
13. Samawi, V.W. Smcsis: An iot based secure multi-crop irrigation system for smart farming. *Int. J. Innov. Comput. Inf. Control* **2021**, *17*, 1225–1241. [CrossRef]
14. Singh, D.K.; Sobti, R.; Kumar Malik, P.; Shrestha, S.; Singh, P.K.; Ghafoor, K.Z. IoT-Driven Model for Weather and Soil Conditions Based on Precision Irrigation Using Machine Learning. *Secur. Commun. Netw.* **2022**, *2022*, 7283975. [CrossRef]
15. Prasanna Lakshmi, G.S.; Asha, P.N.; Sandhya, G.; Vivek Sharma, S.; Shilpashree, S.; Subramanya, S.G. An intelligent IOT sensor coupled precision irrigation model for agriculture. *Meas. Sens.* **2023**, *25*, 100608. [CrossRef]
16. Yazdinejad, A.; Zolfaghari, B.; Azmoodeh, A.; Dehghantanha, A.; Karimipour, H.; Fraser, E.; Green, A.G.; Russell, C.; Duncan, E. A review on security of smart farming and precision agriculture: Security aspects, attacks, threats and countermeasures. *Appl. Sci.* **2021**, *11*, 7518. [CrossRef]
17. Basharat, A.; Mohamad, M.M.B. Security Challenges and Solutions for Internet of Things based Smart Agriculture: A Review. In Proceedings of the 4th International Conference on Smart Sensors and Application: Digitalization for Societal Well-Being, ICSSA 2022, Kuala Lumpur, Indonesia, 26–28 July 2022; pp. 102–107. [CrossRef]
18. Grgic, K.; Pejkovic, A.; Zrnic, M.; Spisic, J. An Overview of Security Aspects of IoT Communication Technologies for Smart Agriculture. In Proceedings of the 16th International Conference on Telecommunications, ConTEL 2021, Zagreb, Croatia, 30 June–2 July 2021; pp. 146–151. [CrossRef]
19. Butun, I.; Österberg, P.; Song, H. Security of the Internet of Things: Vulnerabilities, Attacks, and Countermeasures. *IEEE Commun. Surv. Tutor.* **2020**, *22*, 616–644. [CrossRef]
20. Sadio, O.; Ngom, I.; Lishou, C. Lightweight Security Scheme for MQTT/MQTT-SN Protocol. In Proceedings of the 2019 6th International Conference on Internet of Things: Systems, Management and Security, IOTSMS 2019, Granada, Spain, 22–25 October 2019; pp. 119–123. [CrossRef]
21. Hameed, S.; Khan, F.I.; Hameed, B. Understanding Security Requirements and Challenges in Internet of Things (IoT): A Review. *J. Comput. Netw. Commun.* **2019**, *2019*, 9629381. [CrossRef]
22. Buchanan, W.J.; Li, S.; Asif, R. Lightweight cryptography methods. *J. Cyber Secur. Technol.* **2017**, *1*, 187–201. [CrossRef]
23. Wu, X.W.; Yang, E.H.; Wang, J. Lightweight security protocols for the Internet of Things. In Proceedings of the 2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC), Montreal, QC, Canada, 8–13 October 2018; pp. 1–7. [CrossRef]
24. Lara-Nino, C.A.; Morales-Sandoval, M.; Diaz-Perez, A. An evaluation of AES and present ciphers for lightweight cryptography on smartphones. In Proceedings of the 2016 International Conference on Electronics, Communications and Computers, CONIELECOMP 2016, Cholula, Mexico, 24–26 February 2016; pp. 87–93. [CrossRef]
25. Mishra, B.; Kertesz, A. The use of MQTT in M2M and IoT systems: A survey. *IEEE Access* **2020**, *8*, 201071–201086. [CrossRef]
26. Park, C.S.; Nam, H.M. Security Architecture and Protocols for Secure MQTT-SN. *IEEE Access* **2020**, *8*, 226422–226436. [CrossRef]
27. Hintaw, A.J.; Manickam, S.; Karuppayah, S.; Aboalmaaly, M.F. A brief review on MQTT's security issues within the internet of things (IoT). *J. Commun.* **2019**, *14*, 463–469. [CrossRef]
28. Chen, F.; Huo, Y.; Zhu, J.; Fan, D. A Review on the Study on MQTT Security Challenge. In Proceedings of the 2020 IEEE International Conference on Smart Cloud (SmartCloud), Washington, DC, USA, 6–8 November 2020; pp. 128–133. [CrossRef]
29. RFC.6574 Report from the Smart Object Workshop. 2012; pp. 1–32. Available online: https://www.iab.org/2012/04/12/rfc-6574-report-from-the-smart-object-workshop/ (accessed on 30 December 2022).
30. Amnalou, S.; Bakar, K.A.A. Lightweight security mechanism over MQTT protocol for IoT devices. *Int. J. Adv. Comput. Sci. Appl.* **2020**, *11*, 202–207. [CrossRef]
31. Saeb, M. The Expeditious Cipher (X-cipher): A High Throughput Lightweight Cipher. *Int. J. Comput. Sci. Commun. Secur.* **2014**, *1*, 57–62.
32. Dworkin, M.; Barker, E.; Nechvatal, J.; Foti, J.; Bassham, L.; Roback, E.; Dray, J. *Advanced Encryption Standard (AES)*; NIST: Washington, DC, USA, 2001. [CrossRef]
33. Bogdanov, A.; Knudsen, L.R.; Leander, G.; Paar, C.; Poschmann, A.; Robshaw, M.J.B.; Seurin, Y.; Vikkelsoe, C. PRESENT: An Ultra-Lightweight Block Cipher. In *Cryptographic Hardware and Embedded Systems—CHES 2007, Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems, Vienna, Austria, 10–13 September 2007*; Paillier, P., Verbauwhede, I., Eds.; Springer: Berlin/Heidelberg, Germany, 2007; pp. 450–466.