Design of Authentication Protocols

in

IoT-enabled Smart Agriculture Environment

Thesis submitted in partial fulfillment
of the requirements for the degree of

Doctor of Philosophy

in

Computer Science and Engineering

by

Anusha Vangala

Roll No. 2019801004

anusha.vangala@research.iiit.ac.in

International Institute of Information Technology, Hyderabad
(Deemed to be University)
Hyderabad - 500 032, INDIA

JUNE 2023

International Institute of Information Technology
Hyderabad, India

CERTIFICATE

This is to certify that the thesis entitled **Design of Authentication Protocols in IoT-enabled Smart Agriculture Environment**, submitted by **Anusha Vangala** to International Institute of Information Technology, Hyderabad, is a record of bona fide research work under my supervision and is worthy of consideration for the award of the degree of Doctor of Philosophy in Computer Science and Engineering of the Institute.

Date:

Dr. Ashok Kumar Das
Associate Professor
Center for Security, Theory and Algorithmic Research
International Institute of Information Technology
Hyderabad 500 032, INDIA

Dedicated to my family

Acknowledgments

I extend my appreciation to all the people who have capacitated me in forming an effective thesis from my research work.

First and foremost, I express my deepest gratitude to my esteemed supervisor, Dr. Ashok Kumar Das, for his insightful guidance in steering and molding the research work for this thesis. His rich work ethic has a strong influence on my research experience. The depth of his support and undeviating encouragement is immeasurable. His genuine teachings and feedback have reshaped my research mindset and will stay with me. Words cannot describe the deep respect and regard I carry for him.

I would like to extend my hearty gratefulness to Dr. Sajal Kumar Das, Professor and Daniel St. Clair Endowed Chair, Department of Computer Science, Missouri University of Science and Technology, for his valuable time and constructive criticism on improving my research work.

I would like to thank Dr. Anil Kumar Sutrala, a Principal Software Engineer at CA Technologies – A Broadcom Company, Hyderabad, India, for lending his precious time to help me in my Ph.D. work. I also thank Ankush Mitra, Abhishek Bisht, and Raj Maheshwari from IIIT Hyderabad for their help in the testbed experimental setups.

I deeply thank my mother for her unconditional faith in me, wise advice and for being my strong pillar of support throughout this journey. I thank my son Atharv for being my source of happiness and constantly motivating me into action.

I am grateful to all my lab mates and friends for the intuitive discussions and memorable conversations that contoured my love for research life.

Place : IIIT Hyderabad

Date:

Anusha Vangala
PhD Student
Roll No. 2019801004
Center for Security, Theory, and Algorithmic Research
International Institute of Information Technology
Hyderabad 500 032, INDIA

Abstract

Agriculture is a vital area for the sustenance of humankind, engulfing manufacturing, security, traceability, and sustainable resource management. The agricultural industry has a major contribution to the economy due to its huge share in the gross domestic product (GDP) and as a source of employment. India is currently one of the world's largest producers of agricultural products due to its bio-diversity. With the resources receding expeditiously, it is of utmost significance to innovate techniques that help in the subsistence of agriculture. The past few decades have seen immense change in the operation of the agricultural sector with the introduction of precision farming in conjunction with the Internet of Things (IoT). The growth of IoT and Blockchain technology as two rapidly emerging fields can ameliorate the state of the food chain today. The application of such advancements is highly based on an exchange of messages between various devices in the farming milieu and raises several scenarios which require cryptographic security.

In this thesis, we understand the concept of precision agriculture, its evolution into smart agriculture, and the benefits of such evolution. The applications of IoT in agriculture, which give rise to various developing areas in agriculture, are studied. We study the security scenarios applicable in husbandry through the analysis of possible attacks and threats. The use and evolution of blockchain in the agriculture sector are studied. A layered architecture for smart farming is proposed that is independent of the underlying technologies, and the requirements of cryptographic security have been laid out based on the proposed architecture. A novel generalized blockchain-based security architecture has also been proposed. The testbeds available for IoT-based agricultural systems have been studied in detail. A literature survey of security protocols for various security subsectors in smart agriculture and authentication protocols in various smart applications provides a detailed dissection of the progress in each of the farming security sub-areas. We also perform a rigorous literature review to inspect the state-of-the-art information security using blockchain technology. The current progress in developing IoT-based tools and systems in the industry has also been studied. This research work proposes a series of authentication protocols that address the security issues concerning smart farming, including user authentication and mutual authentication between the various involved entities.

The first contribution of this research work presents a new signature-based three-factor user authentication scheme in intelligent precision agriculture. The established session key between a user and the accessed smart device is then used to communicate securely to fetch real-time data from the device. The proposed scheme relies on one-way hashing and elliptic

curve cryptography (ECC). For user biometric verification, the fuzzy extractor technique has been applied because it is verified using the Hamming distance to avoid false acceptance and rejection errors. A detailed security analysis, including the random-oracle-based formal security, formal security verification using the broadly-recognized Automated Validation of Internet Security Protocols and Applications (AVISPA) tool, and non-mathematical informal security analysis show the robustness of the proposed scheme against a number of potential attacks. In addition, testbed experiments are performed to measure the computational time of various cryptographic primitives used for comparative study among the proposed scheme and other competing schemes. The detailed comparative analysis shows that the proposed scheme has a better trade-off between its offered security and functionality features and communication and computational overheads compared to other competing schemes.

In the second contribution of this thesis, new authentication and key management scheme for IoT-enabled Intelligent Precision Agriculture (IPA), called AKMS-AgriIoT, has been put forward with the private blockchain-based solution. Several IoT smart devices and drones can be deployed in an IPA to monitor an agricultural environment. The drones can be further utilized to collect the data from smart devices and send it to the Ground Station Server (GSS). However, insecure communication among the smart devices, drones, and the GSS make the IoT agriculture environment vulnerable to various potential attacks. The blocks formed with the encrypted transactions and their respective signatures by the GSS are mined by the cloud servers to verify and add the blocks to the private blockchain center. Detailed security analysis and comparative study reveal that the proposed AKMS-AgriIoT supports better security and provides more functionality features, fewer communication costs, and comparable computation costs compared to other relevant schemes. In addition, a blockchain-based implementation of the proposed AKMS-AgriIoT has also been carried out.

The third contribution of this research work involves designing a new smart contract-based blockchain-envisioned authenticated key agreement mechanism SCBAS-SF in a smart farming environment. The device-to-device (D2D) authentication phase and device-to-gateway (D2G) authentication phase support mutual authentication and key agreement between two IoT-enabled devices and between an IoT device and the gateway node in the network, respectively. The edge servers create the blocks on the authenticated sensor data of IoT devices received from the gateway nodes and then sent to the cloud server. The blocks added to the blockchain are a mixture of encrypted and unencrypted sensor data depending on whether it should be available openly to all stakeholders or privately to one particular stakeholder. The blockchain is used to secure sensitive sensor data after authenti-

cation is completed. The smart contract-based consensus mechanism allows verification and addition of the formed blocks by a Peer-to-Peer (P2P) cloud server network. The security of the proposed scheme SCBAS-SF is done through formal and informal security analysis and the formal security verification tool AVISPA. A detailed comparative study reveals that the proposed scheme offers superior security and more functionality features than existing competing authentication protocols. A blockchain-based simulation has also been conducted to measure computational time for a varied number of mined blocks and a varied number of transactions per block. Real-time testbed has been implemented to securely send a captured farm image from an IoT smart device to the cloud server via the gateway node and edge server using the proposed SCBAS-SF protocol.

The fourth contribution of this thesis proposes an efficient blockchain-enabled authenticated key agreement scheme for mobile vehicles-assisted precision agricultural IoT networks, called *AgroMobiBlock*. The limited existing work on authentication in agricultural networks shows passive usage of blockchains with very high costs. In *AgroMobiBlock*, we propose a novel idea using the elliptic curve operations on an active hybrid blockchain over mobile farming vehicles with low computation and communication costs. The formal and informal security analysis along with the formal security verification using the AVISPA software tool have shown the robustness of *AgroMobiBlock* against man-in-the-middle, impersonation, replay, physical capture, and ephemeral secret leakage attacks among other potential attacks. The blockchain-based simulation on large-scale nodes shows the computational time for an increase in the network and block sizes. Moreover, the real-time testbed experiments have been performed to show the practical usefulness of the proposed *AgroMobiBlock*.

Keywords: Intelligent Precision Agriculture (IPA), smart agriculture, smart farming, Internet of Things (IoT), blockchain, Blockchain of Things (BCoT), authentication, key agreement, security, testbed experiments.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The agriculture sector is one of the most influential sectors of India's economy. Being the primary source of livelihood for more than 50% of the population, the agriculture sector's share of gross value added (GVA) among all the sectors is nearly 18% [30]. India is currently one of the world's largest producers of agricultural products due to its bio-diversity. Liu *et al.* [221] presented the emergence of revolutions in the agriculture sector from traditional manual farming practices in the first agriculture revolution, Agriculture 1.0, to the usage of mechanized agricultural machinery in the second agriculture revolution, Agriculture 2.0 and the application of software, communication and information technologies along with embedded systems in the third agriculture revolution, Agriculture 3.0 [109]. Precision agriculture [255, 297, 298, 349], as part of Agriculture 3.0, involves regular monitoring of the yield along with applying automation and information technology to improve the yield [244]. Agriculture 4.0 or smart agriculture or smart farming is an upcoming revolution in the agricultural industry that significantly influences the economy of the country by adopting technologies, such as Internet of Things (IoT), Artificial Intelligence (AI), Wireless Sensor Networks (WSNs), Big data, robotics, and blockchain technology. Monoculture and intensive animal farming are common agriculture production patterns, with mechanization and informatization being the major agricultural production processes. The research challenges of each of the smart technologies applicable in the agriculture sector have been analyzed in detail.

Digitization of the agricultural sector has five themes as follows [196]:

a) Naturalizing the adoption of disruptive digital technologies in the farm environment after assessing the potential uses and benefits

b) Understanding and assessing the effects of digitization on the farm environment and the ease of adoption of such technologies by the farming professionals

c) Evaluating the change in ownership, privacy, power, and ethics of digitization

d) Innovation in the agricultural knowledge systems

e) Evaluating the change in agricultural management and its effect on agricultural economics.

Bacco *et al.* [62] studied various research projects undertaken in the digitization of agriculture for various agricultural operations and technological paradigms in the European Union (EU) territory.

In developing countries, where the economy is based on the agricultural sector, it has been observed that the farming practices are dependent on the ad-hoc intuition and experience of the people involved in farming. This leads to having very minimal control over the amount of produce, and in turn, over the financial gain incurred, even after laborious efforts from the farmers. Strategic techniques to evade such situations are provided by the use of Precision Agriculture (PA) being deployed to monitor and control the approach to smart farming [190, 282, 285].

PA is treated as a mechanism related to farm management in which information technology (IT) plays an important role in assuring that the crops and soil get exactly what they require for maximum health as well as productivity. Therefore, the main purpose of PA is to assure sustainability, profitability, and protection of the environment as well.

Precision farming is an area of research where the concept of IoT is applied with the help of various smart sensor devices, such as ground sensors, radiation sensors, air humidity sensors, optimal sensors, soil moisture sensors, location sensors (GPS), electro-chemical sensors, pH sensors, mechanical sensors, airflow sensors, temperature sensors, accelerometer sensors, gyroscope, smart cameras, and agricultural weather stations placed in the agriculture field that sense various types of conditions in its locality related to seeding, weeding, water study, fertilizer availability, soil study, crop harvest readiness, and disease vulnerability and effect. The perceived data is in raw form and needs processing in order to be useful. Therefore, the collected sensor data goes through filtering and refining processes and converts into functional information. This processed information helps to obtain a comprehensive picture of the state of the agricultural field. It is fed into a decision-making system that analyzes the best possible measure to be taken to improve or sustain the state of the field [145, 167, 190, 276, 282, 285, 306, 317, 352].

PA, also known as "Site-Specific Crop Management (SSCM)", is a technique for automated management of agricultural crops, fields, and animals using a "smart sensing system" which can accommodate an environment that is adapted to the current needs of the system using the disseminated technology. It involves the following steps: a) collecting objective spatial and time-sensitive data using remote and proximal sensors, b) application of filtering methods to extract appropriate data, c) incorporation of the Artificial Intelligence (AI)-specific algorithms for decision making, and d) use of actuation systems to execute the required actions. Such managerial strategies can help to mitigate the issues revolving around food production, distribution, and sustainability [276, 282].

Based on the above understanding of PA, Intelligent Precision Agriculture (IPA)/smart agriculture/smart farming is expected to have the following properties according to Rubio *et al.* [276]: 1) *sensing technologies:* sensing devices could be fixed to autonomous platforms and robots to make observations of the surrounding agriculture environment; 2) *unmanned operations:* it implies the application of robotics and Artificial Intelligence (AI) to replace human workforce with machine workforce; 3) *data-driven:* it involves aggregation of huge amounts of data regarding interesting parameters in the region of interest (ROI); 4) *decision-support system:* an analysis on the parameters is conducted which helps in performing an action in favour of the circumstances in ROI; 5) *actuation technologies:* the decided action needs to be executed either in real-time or deferred-time; and 6) *interoperability of devices:* it involves a network of diverse devices interacting to collect and analyze data.

In the past decade, the IoT has risen into a disruptive technology that can change the face of the current world significantly. IoT is a technology with a number of objects placed in different locations that collaborate collectively to gather data from the surrounding environment. The gathered data is then processed to extract meaningful information that provides an insight into the current state of the system as a whole. This information can then be utilized to be implemented as real-time data in specialized applications. The objects used in such applications can be physical in nature or virtual and have the capability to function without any human intervention [60, 213, 238].

An IoT *connected device* has the ability to sense the surrounding environment to take necessary readings and send the data through the internet to a server that can store the data for future use or to another device, like a smartphone where a user can view the data. This allows continuous monitoring of a system under review. Such a monitored environment allows the user to make decisions on the actions to be performed [324].

A *smart sensing* environment consists of a connected network of devices that can con-

stantly send and receive each other's data. In addition, it has also the capability to take decisions on behalf of a user and perform an action on the environment in order to improve its condition. This change imposed on the environment calls for further monitoring of the surroundings and, thus, continuously moves towards an evolving environment. In a smart sensing system, the user may be sent the monitored data, actions taken, and the effect of actions on the environment. The user may further have the choice to impose an action different from the one determined by the collectively functioning devices with a decision support system. A smart sensing system is required in an agricultural field to assimilate the state of the field, its effect on the growth of the field, and to ascertain the actions required on the field for better outcome of good produce.

1.1 Benefits of smart agriculture

In the following, we discuss several benefits that are achieved through smart agriculture.

- *Quantity of production*: The application of smart technologies in the agricultural sector can help generate a huge increase in the amount of produce generated in the field. This will help provide food for a large population.

- *Quality of produce*: The quality of food produced can majorly affect the health and nutrition of people from various strata in the country. Better quality of food increases the health and lifespan of the population, which helps better contribution of the people towards the economy of the country.

- *Efficiency of agricultural process and usage of resources*: The usage of smart technologies in the regular agricultural processes can improve the efficiency of execution of the processes. This, in turn, promotes better usage of agricultural resources in the process.

- *Optimal cultivation cost*: High quantity, quality, and efficient processes reduce the overall cost of cultivation and, in turn, increase the remuneration of the obtained agricultural output.

- *Reduction of wastage*: Agriculture sector, being one of the largest economic sectors, is responsible for large amounts of wastage of food and other intermediate resources. Smart technologies may be used to monitor and reduce this wastage.

- *Time efficiency*: Smart agriculture has the capability for timely provision of the required pesticides, fertilizers, and other chemicals that can result in timely and qualitative agricultural produce with minimal losses.

- *Environment friendly*: The increased efficiency of agricultural processes and the reduction of agricultural wastage directly decrease the environmental carbon footprint.

1.2 IoT applications in agriculture

The following are some of the major applications of IoT in the agricultural sector:

- *Smart soil cultivation system*: Such a system would perform the pre-harvest preparation of the field soil by plowing, weeding, seedbed preparation, and sowing.

- *Smart irrigation systems*: This system would automate the artificial supply of the required amount of water for plant growth in a controlled manner.

- *Smart fertilizer systems*: This is the process of automating the spraying of fertilizers on the field with control over the quality and quantity of fertilizer and the time period of spraying.

- *Smart pest detection and control systems*: This system monitors and detects infestation of pests, assesses damage to the crops, and also includes techniques to control the infestation.

- *Smart livestock farming*: This involves using smart technologies for breeding livestock and increasing the quality and quantity of the produce with precision agriculture.

- *Smart harvesting system*: This system uses IoT-based techniques to reap the harvest of a field efficiently.

- *Smart farm management system*: Such a system would intend to provide analytics on data to improve the productivity of yield on the field.

- *Smart groundwater quality management system*: The amount and quality of groundwater have a strong influence on the final produce. Therefore, techniques to maintain proper levels of groundwater are applied using IoT in this system.

1.3 Application areas of IoT-based agriculture

The following are the applications areas of agriculture that are developed by using one or more IoT applications as discussed in Section 1.2. Tzounis *et al.* [317] and Talavera *et al.* [306] described the following agricultural areas which require the venture of technology.

1) Agriculture monitoring

To provide an adequate environment for growing crops with the maximum produce, it is crucial to monitor the parameters that affect the growth of plants at every stage of their growth. A number of sensors in both wired and wireless forms, such as ground sensors, climate sensors, weather stations, and radiation sensors, produce data flows, which are stored and used for monitoring, knowledge mining, reasoning, and control. Moreover, any agriculture monitoring involves the following components:

- *Air monitoring*: It involves the collection of parameters, such as temperature, humidity, and pollutants that could alter or damage the crops.
- *Soil monitoring*: It monitors the soil moisture, pH, electric conductivity (EC), and nutrients and chemicals like nitrite present in the soil. The pH in the soil is a very important parameter that reflects if the soil is healthy enough for a good crop. The amount of nutrients appropriate for the crop may either help the crop flourish or damage it.
- *Water monitoring*: It is crucial to monitor the water quality with the pH, temperature, chemicals, and nutrients, measurement of conductivity, turbidity, and also the water level and rainfall in order to provide the right kind of growth-supportive environment.
- *Livestock monitoring*: Sensors that are placed on animals allow to check if any damage is impending on the crop due to animal livestock. In many cases, animal work and products are used to maintain the soil nutrients and promote crops' effective growth. Reynolds *et al.* [269] provided the statistical justification on the need for animals for sustenance of agriculture.
- *Irrigation control*: An automated irrigation system requires the data on groundwater levels and rainfall to minimize water wastage. It also requires monitoring the weather to avoid irrigating the land immediately before or after rainfall.

- *Plant monitoring*: It encompasses studying the plant life closely for any signs of damage due to diseases or bugs periodically so that appropriate action may be taken to circumvent the problem beforehand, if possible, or take recovery measures.

- *Fertilizer and pesticides control*: Regular use of fertilizers and pesticides is among the top exigencies of agriculture. The specific type of fertilizer, the amount required, and the period of time between each spray are decided based on the values of parameters sensed by the various sensors.

- *Illumination control*: Proper sunlight is essential for promoting proper photosynthesis in plants. Ambient light can be controlled through the use of light sensors and actuators.

2) Controlled agriculture/smart greenhouses

Greenhouses provide an artificial environment with proper control over all the required necessities specifically needed for the healthy growth of a crop. The process can be eased for the user by integrating IoT with the use of sensors to supervise the greenhouse.

3) Food supply chain tracking

According to the "Food and Agriculture Organization of the United Nations" [16], the food supply chain consists of the following: 1) pre-harvest agricultural production when the produce is on the farm, 2) post-harvest operations when the produce undergoes basics procedures like cleaning and sorting, 3) secure storage of food, 4) safe transportation of food, 5) food processing when it is made consumable, 6) sale of food and 7) household and business consumption. In all these stages, it is required to monitor and reduce the quality and quantity wastage.

4) Precision farming/smart farming

Based on the analysis of Gebbers *et al.* [145], and Hedley *et al.* [167], precision agriculture can be comprehended as consisting of a collection of technologies, including automated machinery, sensor networks, and data analytics to study and change the dynamic variation in the uncertain parameters of agricultural systems.

1.4 Security in IoT-based agriculture

1.4.1 Threat model

For the security analysis of the schemes reviewed in this thesis, the messages are expected to be transmitted on public channels. For the "Dolev Yao (DY)" model [125], two honest parties aim to transmit messages secretly. The DY model provides a formal security analysis on any generalized model with two honest parties which need to send messages secretly such that no message is dependent on previously sent messages and have access to public keys of the other parties. Such communication can be concurrently held among multiple pairs of parties. The initial knowledge of each party consists of a private key and a public table consisting of the identity and public key of every other party. In such a model, an adversary $\mathcal{A}$ is assumed to have the following capabilities:

- The adversary $\mathcal{A}$ has complete control over the channels used for transmitting messages which allows him/her to capture, remove and alter messages inside the channel.
- The adversary $\mathcal{A}$ can fabricate new messages and circulate them in the channels.
- An adversary $\mathcal{A}$ has the capability to masquerade as one of the honest parties so that the other honest parties cannot detect its true identity.
- An adversary $\mathcal{A}$ can also re-transmit an already transmitted message in the network to the same destination party.
- The adversary $\mathcal{A}$ is stateful while all other honest communicating parties are stateless. It implies that the communicating parties can create new messages only using their initial knowledge and the last received message. On the other hand, an adversary may use recorded messages and stored data in order to create new messages.
- Multiple concurrent executions for the protocol may be created by the adversary $\mathcal{A}$. The honest parties may be involved in more than one such concurrent execution.

In addition, another adversary model called the "Canetti and Krawczyk (CK)" adversary model [94], adds the following capabilities to the adversary $\mathcal{A}$:

- The adversary $\mathcal{A}$ has all the capabilities under the DY model [125].

- The adversary $\mathcal{A}$ can hijack session states for the established sessions between the honest parties. This allows A to obtain any secret keys and secret credentials that have been used during the session between the communicating parties.

In addition, the end-point nodes are assumed to be untrustworthy. Any devices used as part of communication can be physically captured by the adversary $\mathcal{A}$ to apply power analysis attacks [200, 235] and timing attacks [199] to extract secret credentials stored in the devices' memory storage. Obtaining such sensitive data makes the end-points and the system vulnerable to other attacks such as impersonation attacks.

1.4.2 Security requirements

Any security scheme that caters to providing a solution for smart agriculture should focus on one or more of the following security requirements:

- *Authentication*: This attribute encompasses a technique to ensure that an entity is genuine during its identification before providing any access to the system.
- *Integrity*: It is to ensure that information that is received by the receiver is the same as that sent by the sender, without even minor change.
- *Confidentiality*: It is to ensure that information that is sensitive to the system is never disclosed to any party who does not have the authority to its access.
- *Availability*: An entity that has the authorization to access service should never be denied access even under the circumstances of the Denial-of-Service (DoS) attacks.
- *Non-repudiation*: It allows no entity to repudiate the services or actions that were taken up by that entity. This property, in turn, ensures the traceability of a service to an entity.
- *Authorization*: This feature ensures that only an entity that is given the rights to provide any particular network service does so.
- *Freshness*: This property ensures that a message received is not generated before a threshold time period, which disallows any message to be replayed by an adversary.
- *Forward secrecy*: After a node is detached from the network, either voluntarily or otherwise, it should not be allowed access to any communication that continues within the network.

- *Backward secrecy*: A node that has been recently added to the network should not be privy to communication that had taken place before its addition to the group.

1.4.3 Security threats

IoT-enabled smart agriculture can be vulnerable to many possible attacks, and some of them are listed below as discussed in [117].

- *Replay attack*: In a replay attack, during a transmission from an entity, an adversary $\mathcal{A}$ may reuse the content from the previous transmission and attempt to deceive an authorized entity.

- *Man-in-the-middle attack*: During transmission between two entities, $\mathcal{A}$ can read the transmitted messages and may then attempt to modify or delete the contents of the messages delivered to the receiver.

- *Stolen-verifier attack*: If an access point (gateway node) stores a list or a table of passwords corresponding to the identities, $\mathcal{A}$ may attempt to steal the list from the access point, thereby gaining access to all passwords.

- *Stolen/lost smart card attack*: Once $\mathcal{A}$ has obtained a lost/stolen smart card, the techniques such as power analysis attacks [200] and timings attacks [199] can be used to extract the credentials stored in the memory of a smart card or a mobile device. The extracted credentials can then be used to further derive the secret data used in the calculation of the credentials.

- *Password guessing attack*: This attack involves attempts to speculate the correct password using the intercepted messages and illegal access to credentials stored in the smart card or the mobile device.

- *Password change attack*: In this attack, after obtaining access to a stolen smart card or a mobile device, attempts can be made by $\mathcal{A}$ to change the passwords of the existing registered users in order to be able to gain unauthorized access.

- *Denial-of-Service (DoS) attack*: This is a specific type of attack in which services are denied to an authorized user on account of overuse of the system's resources by other factors such as a failure in hardware or software bugs or over-allocation of bandwidth to certain users [338].

- *Privileged-insider attack*: In this kind of attack, an existing user within the system attempts to misuse his/her privileges in order to acquire unauthorized rights to access/modify/delete vital information. To avoid this, the system should have to check if the access provided is legal for the given type of user.

- *Impersonation attack*: If an adversary $\mathcal{A}$ claims to be the sender by sending fake generated messages, he/she is said to impersonate the sender. In such an attack, it is not possible for the receiver to distinguish that the received message is from the adversary and not from the sender.

- *Resilience against sensing (smart) device capture attack*: An adversary $\mathcal{A}$ may physically seize the sensing device and extract sensitive information from the captured device to establish communication with other non-compromised sensing devices. To comprehend the amount of damage caused by a device capture attack, estimations can be made by calculating the probability of compromised communication: a) between two non-compromised devices and b) between a non-compromised device and a user, given that n_c devices are already compromised in a network. The former is considered in the schemes for access control, device authentication, and key management. The latter is considered in a user authentication or a user access control scheme. For any scheme to be resilient to this attack, the effort is to minimize these probabilities. In an ideal case, the probability should be close to zero.

- *Resilience against new sensing devices deployment attacks*: An IoT environment is also susceptible to a number of attacks, such as the illegal deployment of new sensing devices, replication of existing sensing devices, Sybil, and wormhole attacks.

  - In a wormhole attack [169], an adversary tricks two distant nodes into communicating via a wormhole tunnel, using an in-band or out-of-band channel that can circumvent the network traffic, thus deluding the nodes into the impression that they are near to each other. Such a tunnel gives the adversary control over the network traffic. A wormhole attack can lead to other possible attacks on packets, such as modification, sniffing, and dropping.

  - In a *Sybil attack*, the adversary generates multiple pseudonym identities which are misunderstood for multiple entities [126, 247]. This can lead to multiple requests being accepted from the same entity under different identities. The "51% attack

on a blockchain" is an example of a Sybil attack [205]. The identities used by the adversary may already exist in the network or may be newly generated.

– In a *sensing device replication attack* [254], instead of creating fake identities, a compromised node itself is replicated by an adversary deliberately. This allows the adversary to first capture a device, extract sensitive information from it, replicate the information in other nodes, and deploy the replicated nodes in the network to involve them in communication with other non-compromised devices.

1.4.4 Attacks in agriculture

Demestichas *et al.* [120] detailed the cybersecurity attacks, threats, and their mitigation measures that are applicable in the agricultural scenario when combined with smart technologies such as IoT. Sontowski *et al.* [294] studied different types of network attacks on a smart farm, and a Denial-of-Service (DoS) attack, called WiFi deauthentication attack, was launched experimentally on a smart farm architecture in which a Raspberry Pi was forced to disconnect from the network and prevented from reconnecting. Ferrag *et al.* [138] proposed an intrusion detection system against distributed DoS (DDoS) attacks based on deep neural networks, convolutional neural networks and recurrent neural networks. West [336] identified that connectivity and information flow are two enabling factors for digital farms and proposes a framework to predict the vulnerabilities in a digital farming environment. Table 1.1 summarizes various possible attacks categorized by security functionality and their related remedies.

1.4.5 Functionality requirements in agriculture environment

In this section, we list down some basic functionality requirements in an IoT-based agriculture environment.

- *Dynamic new smart device addition*: In an IoT-based agricultural environment, it is extremely essential that a security scheme should support a dynamic node addition facility after the initial deployment of the nodes, including the smart devices in the network. As an IoT smart device in the agriculture environment may be physically captured by an adversary or its battery power may be exhausted, a new device needs to be deployed into the existing network as a replacement.

Table 1.1: Security functionality, attacks, and remedies in smart agriculture categorised by security goals

Goal	Security Functionality	Attacks	Remedies
Confidentiality	Data privacy Location privacy Session key security Perfect forward/ backward secrecy	Location tracking Eavesdropping Data theft/breach Traffic analysis Man-in-the-middle (MiTM) Known-key attack	* Data encryption * Mixing noise * Hiding location
Integrity	Content protection Information reliability	Data pollution/ poisoning Data falsification Data injection Ephemeral Secret Leakage (ESL) MiTM	* Hashing * Message authentication codes
Authentication	User authentication Mutual authentication	Sybil Impersonation MiTM Replay Session hijacking	* Digital signatures * Identity based cryptography * Group signatures * Multi-factor authentication
Availability	Timely accessibility Information usability	Channel interference Physical capture & damage DoS/DDoS Cyberagroterrorism	* Access control * Fault-tolerance
Accountability	Anonymity Traceability	Repudiation Malicious code	* Pseudonyms * Blind signatures Private anonymous channels * Point-to-point channels * Multi-party protocols with unconditional security * Traceable meta-data *Digital signatures

- *High scalability*: Support of high scalability is a basic functionality required in a modern-day IoT-enabled agriculture environment. High scalability should assure that even if the number of IoT smart devices is going to increase, the overall network performance should not be affected by this factor.

- *Mutual authentication*: Among all the security services, mutual authentication and key management are considered two important techniques to assure secure communication in an IoT-enabled agriculture environment. Resource limitations of the IoT smart devices and their vulnerability to physical capture make the design of mutual authentication between two IoT smart devices and also between a remote end-user and an IoT smart device inside the IoT-based agriculture environment become a challenging task.

- *Availability*: The device communication and control in an IoT-based agriculture environment are performed in real-time to keep a real-world impact. For instance, a user (e.g., a farmer) might require to remotely control an agriculture field by monitoring crops condition and also access data at any time.

- *Efficiency*: In an IoT-based agriculture environment, various smart devices are resource-limited, including the limited battery lifetime. In addition, IoT smart devices might also have other constraints, such as storage. Furthermore, the IoT devices might need frequent communication among them for secure communication. As a result, it is desirable that a designed security protocol should encompass minimum possible computation and communication overhead, as well as storage overhead to store the secret credentials, including the session keys for secure communication with other nodes.

1.5 Blockchain technology in smart agriculture

1.5.1 Need for blockchain in smart agriculture

The usage of blockchain technology in smart agriculture is prominent because it enables the storage of sensitive data related to agricultural crop management to be stored securely. The specific solutions based on blockchain technology are very practical in the precision agriculture scenario as they can help to keep extremely sensitive data about agriculture related to the growth and health of harvest and the obtained produce with persistence, auditability, and anonymity. The data from the agricultural fields need to be properly checked for correctness before being stored. Also, once stored, the data cannot be modified as any modification can cause financial discrepancies, which ultimately affect the farmers' livelihood. Therefore, the solutions presented by blockchain technology are widely applicable to practical scenarios that can help the government and other such organizations to plan the provision of food which is a core resource.

Akram *et al.* [49] proposed that the usage of blockchain technology will be invaluable in various areas of smart agriculture, such as 1) *remote monitoring* that allows the stakeholders to monitor various aspects of an agricultural field remotely; 2) *food integrity* which ensures fairness and authenticity of food in the chain at digital layer and physical layer; 3) *resource update* that is the process of checking the distribution of the resource at various stakeholders; 4) *finance management* which ensures that the involved stakeholders obtain their returns appropriately; 5) *remote weather guidance* that can guide the farmers on the management

of crops during different weather conditions; 6) *integrated agricultural problem solving* which refers to the regional and national level agricultural sectors that can be integrated into a single network to solve any problem at both levels. Lin *et al.* [219] proposed a system model that integrates the idea of connecting the national and regional databases. Based on the model, an evaluation tool has been proposed that can determine the need for blockchain technology in the IPA scenario despite the shortcomings of its offerings. They also studied the trade-offs offered by the dynamic nature of blockchain networks in exchange for its limitations.

Ge *et al.* [144] suggested that the agricultural sector is prone to the risk of fraud due to packaged food and beverage companies increasingly needing the proper certification for their products, which can directly affect the health of its consumers. This has to lead to the identification of the need for transparency and trust in the agricultural sector. Blockchain aims to eliminate the participation of third parties in the audits and turn the entire system into a decentralized network promoting food quality, safety, and sustainability. The relevance of using blockchain in the agricultural and food sector has been explored in terms of the opportunities it provides to different stakeholders. Torky and Hassanein [313] studied and highlighted the need and necessity for the use of blockchain in precision agriculture along with the associated challenges and the relevant use cases in extensive detail. They identified that blockchain could solve the security challenges of extending address space, managing the identity of things, and verification of transactions in IoT networks. Specifically, in precision agriculture, blockchain can be used to record data, verify properties, track and monitor movables, link products to tags or codes, and share information about agri-products.

1.5.2 Evolution of blockchain in smart agriculture

Usage of blockchain in smart agriculture can have a considerable positive effect on the economy. A study on the impact of the usage of blockchains in the grains sector in Australia [153] shows an increase in the Gross Domestic Product (GDP) by approximately 2.5%. Christidis and Devetsikiotis [105] showed how the decentralized nature of blockchains can aid the heterogeneity of the Internet of Things, making it widely applicable to numerous domains. A study by the "Food and Agriculture Organization of the United Nations" in conjunction with the "International Telecommunication Union (Bangkok)" in 2019 [14] shows the growing influence of blockchains in various areas of agricultural sectors along with the risks and possibilities it poses. Figure 1.1 shows the evolution of blockchain into smart agriculture.

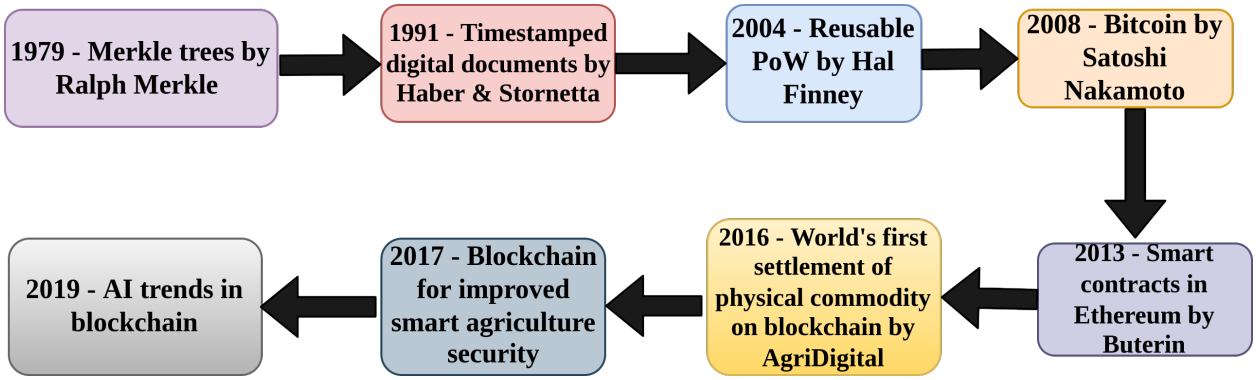

Figure 1.1: Evolution timeline of blockchain in smart agriculture

1.6 Architectures for smart agriculture

1.6.1 Generalized layered architecture

This section proposes a layered architecture that can be generally applied for any implementation of the agricultural environment. The security requirements for the proposed architecture are discussed in detail.

Figure 1.2 shows a generalized layered architecture for smart agriculture.
The proposed architecture for smart farming can be generalized to five layers as follows:

- *Perception layer*: An agricultural environment supported by the Internet of Things (IoT) consists of a network of sensors in a field that have the capability to sense the interesting environmental parameters that help understand the current needs of the crops. The sensors may be of different types: temperature sensors, pressure sensors, light/optical sensors such as infrared sensors, humidity sensors, location sensors, proximity sensors, mechanical sensors such as accelerometers/gyroscopic sensors and motion detectors, image sensors, flow sensors, gas sensors, sound/acoustic sensors, moisture sensors, magnetic sensors, air quality/particulate matter sensors, electro-chemical sensors, water quality sensors such as pH sensors, turbidity sensors, oxidation-reduction potential (ORP) sensors, hydrogen sensors, level sensors, soil sensors, chlorophyll sensor, weather sensors, ultrasonic sensors, vibration sensors and many others [178, 282]. The most appropriate sensor is chosen based on environmental factors such as the power consumption, susceptibility to electromagnetic interferences, temperature and humidity range; economic factors such as the cost, availability, and lifetime along with the characteristics of the sensor such as its sensitivity, range, response time, error rate, etc.

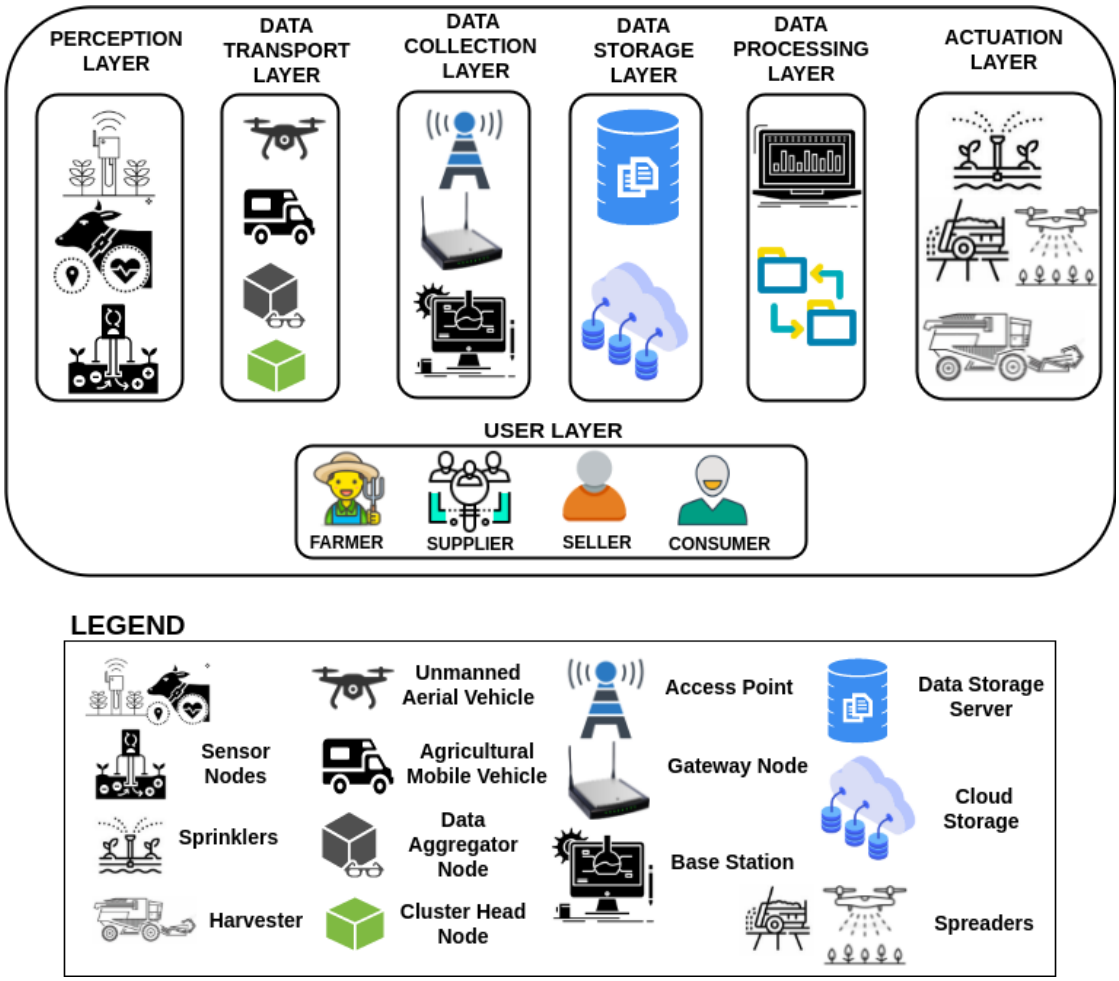

Figure 1.2: Generalized layered architecture for smart agriculture

Moreover, sensors in the agricultural environment are susceptible to extreme weather conditions and rough environments with direct exposure to light, chemicals, and livestock which may damage the sensor. This demands frequent repair or replacement of sensors in the field.

- *Data transport layer*: This layer consists of either movable or immovable devices that are designed to gather data from the sensors in the field. Examples include cluster heads, data aggregation nodes (DAN), unmanned aerial vehicles (UAV), or other agricultural vehicles such as tractors.

- *Data collection layer*: This layer consists of devices such as access points, gateway hubs, or base stations that receive data from the data transport layer with some capability

to group the data meaningfully and prepare it for storage.

- *Data storage layer*: This layer is responsible for the stable storage of the huge amount of data that is received from the various fields. It may include techniques such as database systems, big data storage, or blockchain technology as appropriate.

- *Data processing layer*: This layer consists of techniques to analyze and process the data in order to extract important information that may be fed into a system capable of decision making using artificial intelligence. The results of this layer should be converted into clear instructions compatible with the appliances on the field.

- *Actuation layer*: This layer consists of appliances that can receive and execute the instructions from the data processing layer with correct precision and timing. The precision operation of agricultural machinery consumes energy and requires maintenance. Examples include electric and mechanical linear actuators for forage harvesters, sprinklers, spreaders, seed drills, balers, solenoid valves, heating systems, ventilation systems, condensation systems, humidification, and dehumudification system, shadow tracking system, mechanical and hydraulic systems [178].

- *User layer*: This layer consists of all the stakeholders in the farming process from pre-harvest, production and farming, harvest, storage, transport, and distribution of the final agricultural product. It involves farmers, suppliers, sellers, and consumers.

1.6.2 Architectural security

A thorough understanding of the architecture in Figure 1.2 from the security perspective is discussed by showing the incorporation of security goals into the architecture as follows:

- *Data privacy*: Data from the sensors that are gathered by the devices from the data transport layer need to be secured with algorithms for encryption to ensure that the content of the sensor data is not disclosed to any third-party entity. This will be required for communication between the user layer and perception layer, between the perception layer and data transport layer, between the data transport layer and data collection layer, and between the data collection layer and data storage layer.

- *Location privacy*: The location of the sensors and the immovable devices, along with the real-time location of the movable devices, need to be kept secure in order to ensure

security against damage to the physical device capture attacks and traceability. This security feature will be necessary for the devices in all the layers of the architecture.

- *Entity/user authentication*: It is necessary to ensure the true identity of any entity or user that wants to establish communication with any other entity or user. Both the initiator and responder of the communication must verify themselves before any kind of message exchange takes place. This security goal is required between the user layer and perception layer, between the perception layer and data transport layer, between the data transport layer and data collection layer, and between the data collection layer and data storage layer.

- *Access control*: Once the communicating entity/user is authenticated, it is important to decide the operations that it is allowed to perform in terms of roles, attributes, or privileges using access control policies. This should be applied between all the layers.

- *Authorization*: When an authenticated entity tries to perform an operation within its access control purview, it should be allowed to perform that operation. Otherwise, the system should deny the operation. This should be applied between the user layer and perception layer, between perception layer and data transport layer, between the data transport layer and data collection layer, between data collection layer and data storage layer, and between data processing and actuation layer.

- *Privacy-preserving data aggregation*: This goal should be achieved between the perception layer and data transport layer and between the data transport layer and data collection layer. In data privacy, the individual communication message between devices belonging to different layers is secured, whereas, in privacy-preserving data aggregation, the huge amount of bulk sensor data is to be summarized without revealing its content.

- *Availability*: The sensor data from the field should be available to the authorized entities as and when they request it. In most cases of agricultural scenarios, real-time availability of data is crucial to prevent damage to crop fields. This is an end-to-end goal required by the user layer and perception layer. It may also be required between the data storage and data processing layer. Between the other layers, the need for availability depends on whether the receiver is requesting the data or if the data is periodically sent.

- *Integrity*: The purpose of this goal is to ensure that the content of sensor data is not modified or deleted either intentionally or unintentionally. This goal is required during communication between any of the layers.

- *Secure data search*: Any user at the user layer should be able to perform a search operation over the sensitive sensor data that has been stored in encrypted format at the data storage layer without revealing what is searched or what has been returned as a search result.

- *Non-repudiation*: An action performed by an entity at any of the layers should be recorded well in order to facilitate proper traceability and accountability.

- *Scalability*: Any agricultural environment may need to add more nodes of the same or different sensing capabilities to the field. This requires the network to allow such addition of nodes and increase the overall scalability of the network at any stage. This goal affects the perception layer and the data storage layer as the amount of data to store increases.

- *Forward secrecy*: When a new sensor node is added to the network, it should be restricted to access only the communication that takes place after its arrival into the network. This goal affects only the perception layer.

- *Backward secrecy*: An open agricultural environment has to endure many extreme conditions like the weather changes, usage of equipment and livestock for agricultural activities, infestation by pests and rodents, and other situations that may partly or fully damage the sensor nodes which are part of the smart farming network. In addition, there may be cases when a node is deliberately removed from the network even if it is in proper working condition. In any of these cases, it is important to ensure that once a node is separated from a network, it should not be able to gain access to the secret message exchanges within the network after its departure. This goal only affects the perception layer.

1.6.3 Generalized blockchain-based architecture

A generalized blockchain-based architecture for smart agriculture has been proposed in Figure 1.3.

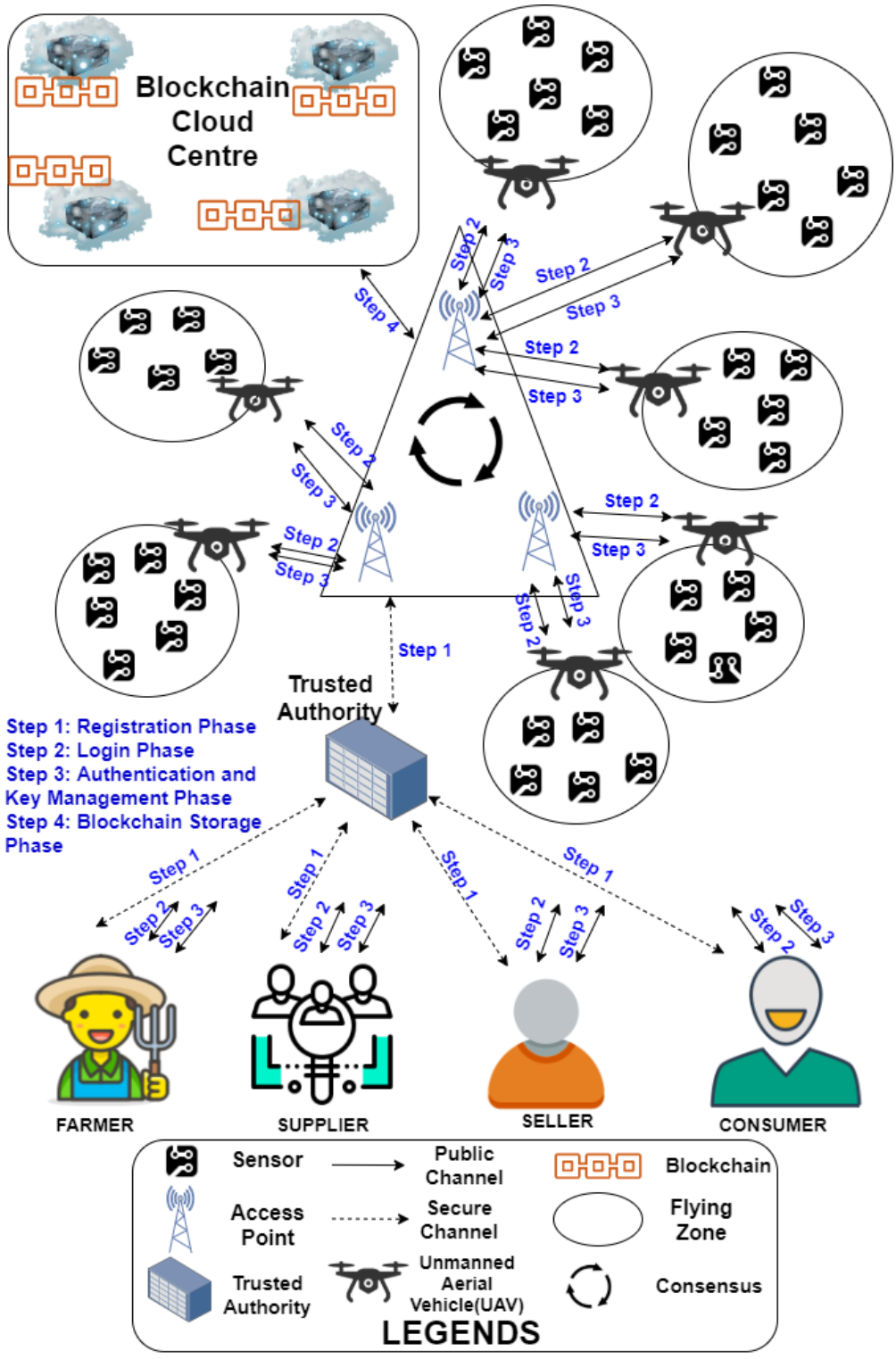

Figure 1.3: Cloud-assisted blockchain-based general architecture for IoT-enabled smart sensing agriculture

An agricultural field may be divided into disjoint zones, called flying zones. Few unmanned aerial vehicles (UAVs), also referred to as drones, may be assigned to a flying zone [233]. The drones collect data from the smart sensor devices placed in their assigned flying zone and forward it to their associated access point. The access points forward the collected data from its associated drones to the user requesting the data. To ensure security in the system, all entities go through a registration phase with a Trusted Authority (TA) before authenticating them for login access to the data. The TA will also assign the sensors (smart devices), drones, and access points in the system. During registration, each of the smart devices, drones, and access points are assigned credentials, including secret keys. After the deployment of the registered entities, a drone needs to authenticate with its access point and also with its other neighbor drones in the flying zone. Similarly, smart devices deployed in the agriculture field need to make secure communication with their neighbor smart devices. To serve these activities, mutual authentication and a key agreement scheme are extremely essential [65]. There are four types of users: a) farmer, b) supplier, c) seller, and d) consumer. Sometimes, an external user (for example, a farmer) can be given access to the real-time data from the deployed smart devices directly. However, to give such access, user authentication is needed where a user will establish session keys with its accessed smart devices provided that they mutually authenticate each other.

The information is collected by the access points from the drones and also by an external user from the accessed smart devices form various transactions. These transactions are then used to form the blocks. Since the information is private and confidential, we consider the private blockchain in the smart agriculture scenario. Note that the transactions are encrypted by the public key of the owner (in this case, an access point), which are then stored in the blocks. The blocks are then mined using some efficient consensus algorithm. For example, we may utilize the Ripple or Practical Byzantine Fault Tolerance (PBFT) consensus algorithm. After successful validation of the blocks by the nodes in the Peer-to-Peer (P2P) network consisting of the trusted access points, the blocks are then added to the blockchain maintained by the blockchain cloud center.

1.7 Motivation and objective of research

The sustainability of humankind is highly dependent on the prosperity of the food and agriculture industry. According to World Population Prospects, 2019 [17], the population will rise globally to 8.5 million by 2030, 9.7 billion by 2050, and 10.9 billion by 2100. According

to the survey report in [18] by the Department of Economic and Social Affairs, United Nations on the world population, the world population could reach 11 billion by the year 2100, with more than 50% concentrated in 9 countries due to a net outflow of more than 1 million in the decade 2010-2020 from the other countries to these nine countries. Even though the population growth rate decreased below 1.1% per year from 2015 to 2020, human life expectancy has increased since 1990 by more than eight years, with an average of 72.8 years per person. The growth of the population increases the demands on the food industry.

The Food and Agriculture Organization of the United Nations published a detailed report [16] on the amount of food that is wasted due to various factors during pre-harvest, harvest, and post-harvest stages, and the depth of its impact on the economy, environment, health and survival of the human race. It uses Food Loss Index (FLI) as an indicative parameter for the percentage of food that is taken off the supply chain. The studies indicate that about 14% of the produced foods are lost by the post-harvest stage globally. The wastage may be in terms of quantitative loss of food leading to a reduced amount of food available for consumption. This is countered using concepts of providing security to food. The other more prevalent form of food wastage is qualitative food wastage which reduces the attributes of food that make it consumable such as the nutritional value of the food, non-compliance with food standards, and their economic impact. Duque-Acevedo *et al.* [128] provided a very deep and thorough analytical study of the food wastage statistics from 1931 to 2018 and the impact of global economic policies on food wastage. They also studied the effect of a new global framework on sustainable development on local policies of different countries. The Food Wastage Footprint–Full Cost Accounting Report [10] by the Food and Agriculture Organization of the United Nations provides a very thorough and detailed study of several statistical methodologies and models that can accurately calculate the cost of wastage of food on various economic and social factors required to sustain the world. It also studies the different factors, natural and otherwise, that affect food wastage directly or indirectly. These statistics suggest the exigency for planning an efficient solution for food production and distribution throughout the world.

The agricultural industry is a significant contributor to the economic growth of any country. The escalating demand for food and its sustenance direct the need for smart farming. The recent multitude of disasters faced since 2020 form an eye-opener on the necessity of precaution needed in the food and agriculture sector. The reports in [5, 142, 161, 261] suggest the effect of the COVID-19 worldwide pandemic on food and agriculture in India and South East Asia stating that 80% of the farms have seen declining sales. Studies by Gregorioa

et al. [152] conclude that the agricultural production has reduced by 17.03 million tons in gross volume in Southeast Asia due to reduced agricultural farm labor during the COVID-19 pandemic affecting 100.77 million individuals. Unprecedented natural disaster incidents such as the locust attack of 2020 in North India [147, 271] where food sufficient for 35,000 people was destroyed by a swarm of 80,000 locusts everyday, and the flooding of Brahmaputra river in Assam [6, 118] can lead to acute nation-wide food crisis.

The lockdown and stoppage of exports during the COVID-19 crisis have led to an excess of agricultural products among farmers. This forces them to sell their products at a very low minimum support price. In India, three new agriculture bills [23, 24, 25] approved in September 2020 promote privatization of the agriculture market such that farmers will be allowed to directly sign contracts with private firms to trade their produce. Such privatization requires an automated system to track the data on produce and trade on top of a smart system that senses and sends accurate field data directly to the concerned stakeholders [91, 134, 177, 263, 318].

The pandemics, disasters, and natural/human-induced calamities directly affect the amount of food produce available to the general population. Along with the quantity of production, the quality of produce plays a vital role in the general health of the public. Ramakumar [264] studied the effect of the Coronavirus disease (COVID-19) pandemic on the agriculture sector worldwide, specifically in India. This study shows that the lack of labor for fieldwork during the lock-down imposed in 2020 resulted in the reduction of total arrival of crops into the agricultural market by 55.6% for wheat, 26.9% for gram, 25.9% for mango, 40.2% for barley of the respective crop amount compared to the produce in 2019.

A remote monitoring system uses sensing and automation technologies to supervise an IoT-based agriculture field. It has great relevance during the work-from-home culture of the pandemic. Remote monitoring systems transmit data related to the soil condition, crops, the effect of used chemicals on the crops, the quantity of yield, quality of yield, and time of yield. These data affect the price of the yield received by the stakeholders. Remote monitoring systems are highly vulnerable to several cyberattacks from unauthorized parties leading to loss of confidentiality, integrity, and data availability.

We now discuss the need for authentication in IoT-based agriculture below. The farming data from a country's numerous states/provinces creates a huge volume of data that becomes a national or global food security concern. Such data is significant in an economic, natural, or human-induced crisis, such as COVID-19 pandemic [1, 3], and political wars [42, 222]. These situations affect a country's economy by forcing its government to increase or de-

crease imports and exports as the need arises. When the national farming data is available to unauthorized entities, such as hostile/rival governments or extremist/radical groups, they are prone to bio-wars that deliberately target the country's fundamental national resource. Hence, there must be proper security protocols to prevent untrusted parties from accessing data circulating in a remote farm monitoring system. Widespread adoption of remote monitoring systems is possible only when they guarantee security against attacks.

Authentication schemes for remotely monitored environments ensure that sensitive information is exchanged only between authenticated parties. A secure remote monitoring system needs to protect the data in transmission and storage from misuse. Such protection of data increases trust in the data. Thus, secure, trusted data promotes informed decision-making to ward off social issues like food crises, low production, and bio-war.

Problem Statement: The transfer of sensor data from a farm to a central storage and processing unit is a core aspect of a smart farming system. Authentication ensures that data is collected only from reliable sources and delivered only to trustworthy entities. This high-level problem statement is fractionalized into the following low-level challenges:

- Sensors must verify themselves securely to an entity requesting their data before establishing an encryption/decryption key in every session,
- Any entity communicating with the sensor must verify its authenticity in every session, and
- Any entity that holds or transmits the sensor data either temporarily or permanently in either encrypted on unencrypted format must authenticate itself before obtaining access to the data.
- A secret key to encrypt the data transmission must be agreed upon in every session.

1.8 Research contributions

The aim of this thesis is to propose a comprehensive suite of protocols to achieve authentication in smart agriculture by considering all the different possible scenarios of the usage of blockchain. The following are the major contributions of this research thesis work.

1.8.1 Signature-based anonymous user authentication in an IoT-enabled IPA environment

We propose a new signature-based three-factor user authentication scheme in an IoT-enabled intelligent precision agricultural (IPA) environment. The proposed scheme is based on "elliptic curve cryptography (ECC)" techniques and signatures. Apart from these, the proposed scheme uses three factors (user password, biometrics, and mobile device) and the widely-accepted fuzzy extractor method [124] for user biometric purposes in order to avoid Denial-of-Service (DoS) attacks as compared to other biometric verification techniques, such as biohashing [69, 184, 223]. The proposed scheme also provides several functionality features, such as the "user mobile device revocation phase" and "dynamic IoT smart device addition phase". The proposed scheme is robust against a variety of potential attacks that are needed in an IoT-enabled intelligent precision agricultural environment, which is shown through the "formal security analysis using the broadly-used Real-Or-Random (ROR) model" [94], the formal security verification using the popular adopted software validation tool known as "Automated Validation of Internet Security Protocols and Applications (AVISPA)" [9] and informal (non-mathematical) security analysis. The testbed experimentation of various cryptographic primitives is performed using the widely-accepted "Multiprecision Integer and Rational Arithmetic Cryptographic Library (MIRACL)" [38] to measure the average time needed for executing the primitives. These experimental results are used in computing the computational time needed for the proposed scheme and other existing competing user authentication schemes. A detailed comparative study of the proposed scheme and other existing competing user authentication schemes shows that the proposed scheme has "a better trade-off among its offered security and functionality features, and communication and computational overheads as compared to those for other competing schemes".

1.8.2 Private blockchain-based authentication in IoT-enabled agriculture

We designed a "new authentication and key management scheme for IoT-enabled IPA, called AKMS-AgriIoT", supported by a private blockchain solution. The sensing data gathered by the drones in the flying zones from the deployed IoT smart devices are securely transmitted to the GSS. The encrypted transactions and their signatures created by the GSS are used by the cloud server(s) for forming blocks, and these are added after verification by "other cloud servers in the P2P CS network" using a consensus algorithm. AKMS-AgriIoT is shown to be

robust against various potential attacks needed in an IoT-enabled IPA through the formal security analysis and informal (non-mathematical) security analysis. Through the formal security verification using the broadly-accepted AVISPA software validation tool, it is shown that AKMS-AgriIoT is also safe against passive/active adversaries. We perform experiments on various cryptographic primitives using the widely-accepted MIRACL library to measure the execution time needed for the cryptographic primitives. A detailed comparative study on "security and functionality features", "communication costs", and "computational costs" among AKMS-AgriIoT and other relevant existing schemes shows the superiority of AKMS-AgriIoT over existing schemes. A blockchain-based implementation of the proposed scheme has been conducted to measure the computational time needed for the varied number of transactions per block and also the varied number of blocks mined in the blockchain.

1.8.3 Hybrid blockchain-based authentication for smart farming using smart contract

We propose a smart contract-based blockchain-envisioned authenticated key agreement mechanism in a smart farming environment. The device-to-device (D2D) authentication phase and device-to-gateway (D2G) authentication phase support mutual authentication and key agreement between two Internet of Things (IoT) enabled devices and between an IoT device and the gateway node in the network, respectively. The formed blocks contain the encrypted data such as the sensing data from the IoT smart devices as well as other unencrypted data (transactions) such as the trading transaction details between farmers and agricultural firms, the chemicals used in the preservation of produce, and the amount and quality of fertilizers used in growing the crops, that may be made available to the buyers of the end-product for deciding among which of the available products to purchase. Thus, a hybrid blockchain is used in the proposed scheme. The voting-based "Practical Byzantine Fault Tolerance (PBFT)" [95] consensus algorithm has been applied with the help of smart contracts to validate and add the blocks into the blockchain using a P2P cloud server network. A detailed security analysis using formal analysis, informal analysis, and formal security verification with an automated software validation tool, AVISPA, has been carried out to show the robustness of the proposed scheme against several potential attacks. A testbed experiment has been done to measure the computational time needed for various cryptographic primitives using the MIRACL library under both a server and a Raspberry PI 3 settings. Next, a detailed comparative analysis shows the proposed scheme

provides superior security and more functionality features and requires low communication costs and comparable computation costs as compared to the existing competing authentication schemes. A blockchain-based implementation has also been carried out for measuring computational time for a varied number of blocks and also a varied number of transactions per block.

1.8.4 Hybrid blockchain-based authentication scheme with mobile vehicles

In this research contribution, the blockchain is leveraged to its full potential by using it in multiple phases. Specifically, we propose a new blockchain-enabled authenticated key agreement scheme for mobile vehicles-assisted precision agricultural IoT networks, called *AgroMobiBlock*, which makes use of an active consortium blockchain. To the best of our knowledge, this is the first attempt to use an active blockchain alongside elliptic curve operations in smart farming. The proposed scheme also leverages vehicular farming systems during the authentication process, which has not been explored in the existing farming applications. A detailed formal security analysis under the widely recognized ROR model" [43], informal security analysis and also formal security verification with the help of automated software tool, known as AVISPA [9] shows that AgroMobiBlock is highly robust against various potential attacks. Next, the textbed experiments for server and Raspberry PI 3 settings using the "MIRACL Cryptographic SDK: Multiprecision Integer and Rational Arithmetic Cryptographic Library" [38] have been performed to measure the computational time required for different cryptographic primitives. A detailed comparative study shows that AgroMobiBlock offers better security and more functionality features, and comparable communication and computational costs as compared to other competing schemes. A real-time implementation using testbed setup gives the step-wise execution time of each phase of *AgroMobiBlock*. In addition, the blockchain simulation observes that the consensus time has a significant increase with the number of nodes and a small increase with the number of transactions. Increasing the number of transactions per block increases the throughput, but it reduces the service time of the blockchain as well.

1.9 Organization of the thesis

The thesis is organized as follows.

In **Chapter** 1, we begin with the digital developments in agricultural sector and its security impact in detail. It weighs the benefits of smart agriculture and studies agricultural areas where IoT can be applied, applications that IoT can support in the agricultural sector, the threat models applicable for IoT-based agriculture, the security requirements, threats, attacks, and functional requirements in smart agriculture. We then look at the blockchain outlook on smart agriculture. We propose two architectures - a layered architecture with a security perspective and a blockchain-based architecture. Then, we present the motivation for our research.

In **Chapter** 2, we present the mathematical concepts that form the basis of the proposed authentication protocols. It includes biometric fuzzy extraction, elliptic curve cryptographic operations, one-way hashing, and blockchain technology. In addition, it also gives an overview of the tools used for the practical implementation of the protocols, such as AVISPA, Hyperledger fabric, Node.js, and MIRACL library.

In **Chapter** 3, we present an extensive literature survey of smart agriculture. It first looks at existing literature surveys and available testbeds. It is followed by a study of the existing security protocols for various security sub-sectors in smart agriculture. Then we study the state-of-the-art authentication schemes that are independent of blockchain technology followed by the blockchain-based authentication schemes, along with their thorough comparative analysis on the basis of computational cost, communication cost, and supported security features.

In **Chapter** 4, we propose a novel user authentication scheme for IoT-enabled precision agriculture system.

In **Chapter** 5, we propose a mutual authentication scheme that uses a private blockchain over an agricultural network with unmanned aerial vehicles (UAV).

In **Chapter** 6, we propose a mutual authentication scheme that uses a hybrid blockchain with the aid of smart contracts.

In **Chapter** 7, we propose a mutual authentication scheme that uses hybrid blockchain (both passive and active blockchains) and mobile farming vehicles without the aid of smart contracts.

In **Chapter** 8, we present the conclusion of our research work and the possible future research directions that may be explored.

Chapter 2

Conceptual Foundations

This chapter covers the concepts of one-way hash functions, elliptic curve cryptography, and biometric recognition that form the cryptographic and mathematical foundation for the authentication schemes designed from Chapter 4 to Chapter 7. In addition, we discuss the blockchain technology used in Chapters 5, 6 and 7. We discuss the tools and libraries that were used to verify and simulate the designed schemes, such as "Automated Validation of Internet Security Protocols and Applications (AVISPA)", "Multiprecision Integer and Rational Arithmetic Cryptographic Library (MIRACL)", Node.js, and hyperledger sawtooth in Appendices A, B, C, D.

2.1 One-way hash functions and their properties

A cryptographic hash function is a one-way function that takes an input data of variable length and produces an output data of a fixed length. A one-way function can be computed to obtain the output in polynomial time. However, it cannot be inverted to obtain the input from the given output in polynomial time. The output of this hash function is called the hash or message digest.

More precisely, a hash function $h : I \rightarrow O$ is a deterministic mapping from a set of strings $I = \{0,1\}^*$ of a long arbitrary number of bits to a set of strings $O = \{0,1\}^{len}$ of decided length of len bits, referred to as the hash or message digest. The hash values generated are evenly distributed over the output set O and hence, appear randomly. This randomness leads to the one-way property that makes it difficult to inverse map the hash value from set O to its appropriate input from set I. This property ensures that collisions of the hash values are minimal, that is, two or more hash outputs cannot be mapped to the same input.

A collision is mathematically described as follows: for any two inputs i_1, $i_2 \in I$ such that $i_1 \neq i_2$, but $h(i_1) = h(i_2)$.

Definition 2.1 (One-way collision-resistant hash function)**.** *A one-way collision resistant hash function* $h : \{0,1\}^* \rightarrow \{0,1\}^{len}$ *is a deterministic algorithm that takes an input as an arbitrary length binary string* $x \in \{0,1\}^*$ *and produces a fixed-length binary string, say len bits string,* $h(x) \in \{0,1\}^{len}$*. The formalization of an adversary* $\mathcal{A}$*'s advantage is*

$$Adv_{\mathcal{A}}^{HASH}(t) = Pr[(x, x^{'}) \leftarrow \mathcal{A} : x \neq x^{'} \, and \, h(x) = h(x^{'})],$$

where $Pr[E]$ *denotes the probability of an event* E*, and* $(x, x^{'}) \leftarrow \mathcal{A}$ *denotes the pair* $(x, x^{'})$ *is randomly selected by* $\mathcal{A}$*. In this case,* $\mathcal{A}$ *is also allowed to be probabilistic and the probability in the advantage is computed over the random choices made by* $\mathcal{A}$ *with the execution time* t*. By an* (ϵ, t)*-adversary* $\mathcal{A}$ *attacking the collision resistance of* $h(\cdot)$*, the runtime of* $\mathcal{A}$ *is bounded by* t *and that* $Adv_{\mathcal{A}}^{HASH}(t) \leq \epsilon$*.*

Applications of cryptographic hash functions include integrity verification of the message on which the hash value is generated using digital signatures, message authentication code (MAC), faster search using dictionaries, and generating data fingerprint to avoid redundancy. Hash functions can also be used as random oracles, in constructing the Merkle trees and in designing commitment schemes.

2.1.1 Properties of cryptographic hash functions

Cryptographic hash functions demonstrate the following set of properties:

- *Easy to compute:* Given an input $i \in I$, computation of $h(i)$ can be completed in polynomial time.
- *Deterministic:* Given an input $i \in I$, $h(i)$ always produces the same message digest.
- *Avalanche effect:* Even a slight change in one or two bits of the input $i \in I$ leads to changes in many bits in unpredictable positions during the hash computation, leading to a very different hash output.
- *One-way property/pre-image resistance:* Given a hash message digest value $o = h(i) \in O$, it is computationally infeasible to map it to its precise input string i from the input set I.

- *Weak collision property/second pre-image resistance:* Given any input string $i_1 \in I$ having a corresponding message digest $o_1 = h(i_1) \in O$, it is computationally infeasible to find another input $i_2 \in I$ whose hash computation $o_2 = h(i_2)$ collides with o_1, that is, $o_2 = h(i_2) = o_1 = h(i_1)$.

- *Strong collision resistance:* For any two inputs i_1, $i_2 \in I$ such that $i_1 \neq i_2$, it is computationally infeasible to find that $h(i_1) = h(i_2)$.

The size of the message digest generated by a hash function is directly proportional to the number of operations that are used in hash computation. An *len*-bit message digest requires 2^{len} operations to detect a weak collision and $2^{len/2}$ operations to detect a strong collision using a third birthday attack. It is computationally infeasible for a message digest of size *len* greater than 80 bits to find a weak collision and *len* greater than 160 bits to find a strong collision by brute force.

2.1.2 Some cryptographic hash algorithms

Cryptographic hash algorithms use the Merkle Damgard scheme as the underlying model for their design. The design of hash algorithms requires the use of compression functions. A compression function may be designed and inserted in the Merkle Damgard scheme. These compression functions may be fully custom-designed or maybe a symmetric key block cipher. The compression function may be designed in two ways: iterated hash function or sponge construction. An iterated hash function uses a compression function to reduce an n-bit string to a m-bit string with $n > m$. A sponge construction consists of a bijective mapping function that maps fixed bit-sized strings to strings of the same size. It pads the input string to make it a multiple of a block length b and splits the padded message into blocks of size b bits. A Merkle Damgard hash model is an iterated hash function that is collision-resistant if the underlying compression function is collision-resistant [140, 302].

The most widely used hashing algorithms for cryptographic purposes with fully custom-designed compression functions are:

(1) *MD Hash:* The hash function in the Message Digest (MD) family are iterated hash functions based on the Merkle-Damgard scheme. The MD hashing consists of message expansion, followed by consecutive rounds of similar steps, ending with the chaining of input to output to complicate inversion. MD2, MD4, and MD5 are the schemes designed under the MD family by Ron Rivest and produce 128-bit hash output. MD2 is not secure

for practical use anymore. MD4 is vulnerable to a collision attack. MD5 is widely used in checksum computation to verify data corruption despite its flaws.

(2) *Secure Hash Algorithm (SHA):* The Secure Hash Algorithm consists of families of algorithms based on the Merkle Damgard scheme. SHA-0, SHA-1, and SHA-2 are iterated hash functions. SHA-0 and SHA-1 produce 160-bit hash in 80 rounds with 512-bit block size using the operations AND, OR, XOR, circular left rotation, and 32-bit modulo addition. SHA-0 is now obsolete due to collisions found in 2^{61} steps. A collision attack with 2^{80} steps that is 10^5 times faster than brute force search birthday paradox was identified against SHA-1 in 2017.

SHA-2 is a family of four hash functions: SHA2-224, SHA2-256, SHA2-384, and SHA2-512, which produce 224-bit, 256-bit, 384-bit, and 512-bit hashes, respectively. SHA2-384 and SHA2-512 use the operations AND, OR, XOR, rotation, shift right, and 64-bit modulo addition with 1024-sized blocks in 80 rounds. SHA2-224 and SHA2-256 use 32-bit addition modulo instead of 64-bit modulo addition with 512-bit blocks in 64 rounds. Iterated hash functions do not allow the use of hash algorithms as keyed hash functions and thus are inapplicable for Message Authentication Codes (MAC).

SHA-3 uses sponge construction with input and output lengths of b bits, which is the sum of the bit rate r and the capacity c. The message is processed r bits at a time, affecting the efficiency of the sponge function. The hash function has a security level of $2^{c/2}$ against collision attacks. The security of the sponge function cannot be higher than that of a random oracle that produces a c bit output. SHA-3 family consists of SHA3-224, SHA3-256, SHA3-384, SHA3-512, SHAKE128, SHAKE256, cSHAKE128, and cSHAKE256, which are based on the Keccak sponge function [4]. SHA3-224, SHA3-256, SHA3-384, and SHA3-512 produce output hashes of sizes 224-bits, 256-bits, 384-bits, and 512-bits using block sizes of 1152 bits, 1088 bits, 832 bits, and 576 bits in 24 rounds respectively. SHAKE128, cSHAKE128, SHAKE256, and cSHAKE256 produce hashes of unlimited length using block sizes of 1344 bits for SHAKE128, cSHAKE128 and 1088 bits for SHAKE256, cSHAKE256 in 24 rounds. All the SHA-3 functions use AND, XOR, NOT and rotation operations [302]. The security level of SHA3-224, SHA3-256, SHA3-384, and SHA3-512 is 112, 128, 192, and 256 bits, respectively, while that of SHAKE128, cSHAKE128 is 128 bits and SHAKE256, cSHAKE256 is 256 bits.

(3) *Others:* RIPEMD-160 produces a hash of 160 bits using dual executions of MD5 parallely. HAVAL allows hashes of variable sizes 128, 160, 192, 224, and 256 bits using 1024

bit block size [140].

There are hashing schemes that use symmetric block ciphers as the compression function. Rabin scheme replaces any symmetric block cipher in the Merkle-Damgard iterated hash function. Davies-Meyer scheme adds forward feed to the Rabin scheme and secures it against Man-in-The-Middle (MiTM) attacks. Matyas-Meyer-Oseas scheme uses the input message itself as the key in the Davies-Meyer scheme. Miyaguchi-Preneel scheme adds bitwise EXORing of the plaintext, ciphertext, and the key to derive the resultant digest in the Matyas-Meyer-Oseas to give stronger immunity against attacks [140].

2.2 Elliptic curve cryptography

Elliptic curves are a set of solutions that satisfy equations of the forms $y^2 = E(x, y)$ where $E(x, y)$ is a cubic polynomial in x and y. A singular elliptic curve with repeated roots of the elliptic curve equation can make the discrete logarithm problem trivial at the singular points given by the roots. Hence, we always choose a non-singular/smooth elliptic curve i.e.one which satisfies the equation $4a^3 + 27b^2 \neq 0$, for cryptographic purposes [330].

In cryptography, we consider elliptic curves as the set of all points that satisfy the Weierstrass Normal form along with the point at infinity. The Weierstrass Normal form is defined as $y^2 = x^2 + ax + b$, where the coefficients a, b lie in the field K, whose characteristic is not 2 or 3. The point at infinity, O, is a point above the y-axis through which a line passes when it is vertical, i.e., $x = 0$. The y-axis is said to be wrapping around, leading to a single point at infinity considered as both the top and bottom of the y-axis, through which every vertical line intersects. The Weierstrass elliptic curve is said to be symmetric about the x-axis. That is, for given values of a and b, the curve includes points for both positive and negative values of y for each given value of x [287].

A "non-singular elliptic curve $E_q(a, b)$ of the type: $y^2 = x^3 + ax + b \pmod q$ over the Galois field $GF(q)$, with a point at infinity (zero point) $\mathcal{O}$, constants $a, b \in Z_q = \{0, 1, 2, \cdots, q-1\}$ such that $4a^3 + 27b^2 \neq 0 \pmod q$" is satisfied, q is a sufficiently large prime so that the computational "elliptic curve discrete logarithm problem (ECDLP)" and "elliptic curve decisional Diffie-Hellman problem (ECDDHP)" become intractable. A base point $G \in E_q(a, b)$ is defined whose order n_G is as large as q, that is, $n_G \cdot G = G + G + \cdots + G$ (n_G times) $= \mathcal{O}$, the point at infinity or zero point, where $n_G \cdot G$ represents the elliptic curve point (scalar) multiplication. For all points on elliptic curve $P = (x_P, y_P)$, $Q = (x_Q, y_Q) \in E_q(a, b)$, the following properties are satisfied [198]:

i) $P + \mathcal{O} = \mathcal{O} + P = P, \forall\ P \in E_q(a, b)$.

ii) $-P = (x_P,\ -y_P) \in E_q(a, b)$ is the additive inverse of $P = (x_P,\ y_P)$ and is called the image of P. Then, $P + (-P) = \mathcal{O}$.

iii) The number of points N in $E_q(a, b)$ is approximately equal to the number of elements in Z_q as given by Hasse's bound [299]:

$$q + 1 - 2\sqrt{q} \leq N \leq q + 1 - 2\sqrt{q}.$$

2.2.1 Elliptic curve arithmetic

- *Elliptic curve point addition*: Given two points P and Q, the point $R = P + Q = (x_R, y_R)$ is determined by the following formula:

$$x_R = (\lambda^2 - x_P - x_Q) \pmod q$$

$$y_R = (\lambda(x_P - x_R) - y_P) \pmod q,$$

where

$$\lambda = \begin{cases} \frac{y_Q - y_P}{x_Q - x_P} \pmod q, & \text{if } P \neq -Q \\ \frac{3x_P^2 + a}{2y_P} \pmod q, & \text{if } P = Q. \end{cases}$$

Geometrically, for two distinct points P and Q, a line passing through the points P, Q is extrapolated to meet the curve at a third point R, whose image is considered the result of the point addition. For a point, P, the tangent on the point P intersects the elliptic curve at another point R, whose image is considered as the result of the point doubling.

- *Elliptic curve point multiplication*: Scalar multiplication of an elliptic curve point $P = (x_P,\ y_P)$ with a scalar $k \in Z_q^* = \{1, 2, \cdots, q-1\}$ denoted as $k{\cdot}P$ is defined by using the repeated additions as $k{\cdot}P = P + P + \cdots + P$ (k times). The number of additions to be performed can be further optimized by using point doubling operations.

Now, we look at the computationally hard problems that make the elliptic curves highly secure for usage in cryptography.

2.2.2 Elliptic curve discrete logarithm problem (ECDLP)

Consider a finite cyclic group Z_q^* with order $q-1$, a primitive element $\alpha \in Z_q^*$ and another element $\beta \in Z_q^*$, the generalized Discrete Logarithm Problem (DLP) can be defined as the problem of determining the value of the integer n with $1 \leq \mathrm{n} \leq q-1$ such that

$$\alpha^n = \beta \pmod{q}.$$

The "elliptic curve discrete logarithm problem (ECDLP)" is defined as follows. Given the points $P \in E_q(a,b)$ and $Q = k \cdot P \in E_q(a,b)$ in a non-singular elliptic curve $E_q(a,b)$, to find the discrete logarithm $k \in Z_q^*$.

Definition 2.2. *Consider a non-singular elliptic curve $E_q(a,b)$ modulo a prime q, with $P \in E_q(a,b)$ and $Q = k \cdot P \in E_q(a,b)$ be two points, where k is chosen randomly from Z_q^*.*
***Instance**: (P, Q, m) for some $k, m \in Z_p$.*
***Output**: Yes, if $Q = m \cdot P$, i.e., $k = m$, and No, otherwise.*
Consider the following two probability distributions:

$$\nabla_{real} = \{k \in Z_q, U = P, V = Q(= k \cdot P), W = k : (U, V, W)\},$$

$$\nabla_{random} = \{k, m \in Z_q, U = P, V = Q(= k \cdot P), W = m : (U, V, W)\}.$$

The advantage of any probabilistic polynomial-time (PPT), 0/1-valued distinguisher ∇ in solving ECDLP on $E_q(a,b)$ is defined as:

$$\begin{aligned} Adv^{ECDLP}_{\nabla, E_p(a,b)} &= |Pr[(U,V,W) \leftarrow D_{real} : \nabla(U,V,W) = 1] \\ &\quad - Pr[(U,V,W) \leftarrow D_{random} : \nabla(U,V,W) = 1]|, \end{aligned}$$

*where the probability $Pr(\cdot)$ is taken over the random choices of k and m. ∇ is called an (t,ϵ)-ECDLP distinguisher, if ∇ runs at most in time t with $Adv^{ECDLP}_{\nabla, E_p(a,b)}(t) \geq \epsilon$. The **ECDLP assumption** states that there exists no (t,ϵ)-ECDLP distinguisher for $E_q(a,b)$. Thus, for every ∇, $Adv^{ECDLP}_{\nabla, E_q(a,b)}(t) \leq \epsilon$ for any sufficiently small $\epsilon > 0$. Currently, there exist only algorithms like Pollard Rho and Baby-step-giant-step that solve ECDLP in exponential and sub-exponential time. ECDLP cannot be solved in polynomial time with the existing algorithms in literature.*

2.2.3 Elliptic curve Diffie-Hellman problem (ECDHP)

Let $E(Z_q)$ be the set of points of an elliptic curve E in a finite field Z_q, where $Z_q = \{0, 1, 2, \cdots, p-1\}$. Given three points P, $a \cdot P$ and $b \cdot P$ on the curve $E(Z_q)$, the Elliptic Curve Diffie-Hellman Problem (ECDHP) requires the computation of $ab \cdot P$, which is

used as the shared secret key between the two communicating parties. The parties only need to communicate the x-coordinate, which can be used to determine the y-coordinate by computing the square root in Z_q. In cryptographic circles, the idea of sending the x-coordinate plus one extra bit is known as point compression.

1) *Elliptic curve decisional Diffie Hellman problem (ECDDHP)*: Let P be a point in $E_q(a, b)$. Considering that the points in the quadruple (P, $\gamma_1 \cdot P$, $\gamma_2 \cdot P$, $\gamma_3 \cdot P$) are known, the ECDDHP is to decide whether $\gamma_3 = \gamma_1 \cdot \gamma_2$ or a uniform value, where γ_1, γ_2, $\gamma_3 \in Z_q^*$.

2) *Elliptic curve computational Diffie Hellman problem (ECCDHP):* Let P be a point in $E_q(a, b)$. Considering that the points $\gamma_1 \cdot P \in E_q(a, b)$ and $\gamma_2 \cdot P \in E_q(a, b)$ are known where γ_1, $\gamma_2 \in Z_q^*$, the ECCDHP states that it is computationally infeasible to compute $\gamma_1 \gamma_2 \cdot P$.

For a prime q greater than 160-bits, the ECDLP, ECCDHP, and ECDDHP are considered intractable, making ECC a highly secure option to be used in cryptography. In addition, elliptic curve cryptography allows about 5 to 10 times smaller key sizes as compared to RSA or Z_q^* based DLP cryptographic designs.

2.2.4 Elliptic curve encryption and decryption

ElGamal scheme is considered very close to the security of the discrete logarithms for the group $E(Z_q)$. This section presents the elliptic curve variant of the ElGamal scheme for encryption and decryption [299].

Step 1 *(Initialization):* To initiate the algorithm, Alice and Bob have to agree on the following parameters: a finite field Z_q, an elliptic curve $E_q(a, b)$, and a base point G of the prime order n_G. The following parameters are made public: $\{Z_q, E_q(a, b), G, n_G\}$.

Step 2 *(Key generation):* Alice chooses a private key $p_A \in Z_q^*$ and computes her public key $P_A = p_A \cdot G$. Similarly, Bob chooses a private key $p_B \in Z_q^*$ and computes his public key $P_B = p_B \cdot G$.

Step 3 *(Encryption):* For a message M, Alice must first embed it into the elliptic curve and obtain a message point P_M. To encrypt the plaintext point P_M, Alice chooses a random positive integer $a_e \in Z_q^*$ and computes the coordinates of the ciphertext

point C_M using the receiver Bob's public key P_B as

$$\begin{aligned} C_M &= \{a_e \cdot G, P_M + a_e \cdot P_B\} \\ &= (C_1, C_2). \end{aligned}$$

Step 4 *(Decryption):* To decrypt the plaintext point P_M, Bob multiplies the first coordinate in received ciphertext point C_M with its own private key p_B and subtracts it from the second coordinate as follows:

$$\begin{aligned} C_2 - p_B C_1 &= P_M + a_e \cdot P_B - p_B(a_e \cdot G) \\ &= P_M + a_e(p_B \cdot G) - p_B(a_e \cdot G) \\ &= P_M. \end{aligned}$$

If T_{ecm} and T_{eca} denote the time taken by the elliptic curve multiplication and elliptic curve addition operations, respectively, the elliptic curve encryption needs the time complexity of $2T_{ecm} + T_{eca}$ and decryption takes the time complexity of $T_{ecm} + T_{eca}$. The resultant of encryption consists of two elliptic curve points of 320 bits each. Thus, the final ciphertext C_M has a total size of $(320 + 320) = 640$ bits.

2.2.5 Elliptic curve digital signature algorithm (ECDSA)

The "Elliptic Curve Digital Signature Algorithm (ECDSA)" [186] is a variant of the ElGamal signature scheme and is considered the elliptic curve equivalent of the "Digital Signature Algorithm (DSA)". Consider that Alice has to sign a message M, that needs to be verified by Bob. The ECDSA has the following steps:

Step 1 *(Initialization)*: To initiate the algorithm, Alice and Bob have to agree on the following parameters: a finite field Z_q, an elliptic curve $E_q(a, b)$, and a base point G of the prime order n_G.

Step 2 *(Key generation)*: Alice chooses a private key p_A and computer her public key $P_A = p_A \cdot G$. Alice publicly announces $\{Z_q, E_q(a, b), G, n_G, P_A\}$.

Step 3 *(Signing)*:

a) For a message M, Alice first calculates its hash $m = h(M)$, which returns the hash value m.

b) Alice chooses a statistically random unique integer $a_s \in Z_q^*$ and computes $A_s = a_s \cdot G = (x_a, y_a)$. It checks if $x_a \neq 0 \pmod{q}$ and assigns $s_1 = x_a$. If $x_a = 0$, Alice chooses another $a_s \pmod{q}$ and repeats Step 3b.

c) Alice computes $s_2 = (m + p_A s_1) a_s^{-1} \pmod{q}$. If $s_2 = 0$, Alice chooses another $a_s \pmod{q}$ and repeats Step 3c.

Alice sends the signature (s_1, s_2) along with the message M to the verifier, Bob.

Step 4 *(Verification)*:

a) For the message M, Bob first calculates its hash $m = h(M)$, which returns the hash value m.

b) Bob computes $b_1 = m s_2^{-1} \pmod{q}$ and $b_2 = s_1 s_2^{-1} \pmod{q}$.

c) Bob computes the point $b_1 \cdot G + b_2 \cdot P_A = (x_b, y_b)$ on elliptic curve $E_q(a, b)$ using the signer Alice's public key P_A and the base point G.

d) Bob then extracts $v = x_b \pmod{q}$ and verifies if $v \stackrel{?}{=} s_1$. If this is true, the signature is accepted.

If T_h, T_{ecm} and T_{eca} represent the time taken for the "hash", "elliptic curve point multiplication" and "elliptic curve point addition" operations, respectively, the signing will take the time complexity of $T_h + T_{ecm}$, whereas the verification will need the time complexity of $2T_{ecm} + T_{eca} + T_h$. The resultant signature consists of two integers s_1 and s_2, each of 160 bits. Thus, the final signature has a size of $(160 + 160) = 320$ bits.

2.3 Biometric recognition

Biometric recognition is the field of study on recognizing an individual using his/her/their inherent biological traits from birth. A biometric system consists of an enrollment phase that uses image acquisition and feature extraction, and a comparison phase that uses a matching algorithm and database storage. The comparison phase in biometric recognition may be performed in one of three modes:

(1) Verification mode compares one entered biometric datum with one stored biometric data for each identity. It is a positive recognition mode that prevents the same identity from being reused by multiple users.

(2) Identification mode compares one entered biometric datum to all the stored biometric data for each identity. It is a negative recognition mode to prove that a person is whom they deny being.

(3) Screening mode compares one entered biometric datum to a subset of the stored biometric data for each identity. This is also a negative recognition mode [176].

2.3.1 Biometric attributes

A biological attribute/trait can be considered appropriate for biometric recognition if it has the following properties [259]:

(1) *Universality:* All the beings that use the system must possess this attribute.

(2) *Distinctiveness:* The attribute must be unique to every being.

(3) *Permanence/invariance:* The attribute must remain invariable over time.

(4) *Recordability:* The attribute can be recorded, stored, and retrieved in an appropriate format.

(5) *Acceptable:* People must be willing to submit the attribute.

(6) *Circumvention:* This property shows the level at which a particular biometric attribute may be misused.

(7) *Performance:* This property refers to the accuracy in associating an attribute to its correct owner and the efficiency of this process.

Matching the biometric attribute involves the usage of pattern recognition techniques. The possible physical and physiological attributes that can be used as biometric attributes are "deoxyribonucleic acid (DNA) which is an organic chemical that contains genetic information and instructions for protein synthesis", ear, face, hand, fingerprint, palmprint, face, and hand geometry, hand vein, finger knuckle, teeth, tongue print, iris, odor, heart sound, retina, and voice. Gait, signature, and keystroke are some behavioral traits. Ethnicity, scars, tattoos, marks, and height are some of the soft traits which lack distinctiveness and permanence properties. However, a unimodal characteristic cannot guarantee 100% accuracy. Therefore, it is suggested to use multi-modal biometrics to collect multiple traits either in a serial/cascaded or parallel fashion to establish identity with high accuracy [319].

There are three types of errors possible in biometric recognition:

(i) False acceptance, where an imposter is accepted as an authorized user, measured using False Acceptance Rate (FAR).

(ii) False rejection, where an authorized user is deemed as an imposter and blocked from the system, measured using False Rejection Rate (FRR).

(3) Equal error, where the FAR and FRR are equal, measured by Equal Error Rate (EER).

A trade-off between FAR and FRR is necessary as an increase in FAR reduces FRR. So, an optimal threshold is applied to the FAR and FRR values. An ideal biometric system would have an EER of zero.

2.3.2 Techniques for biometric recognition

There are three major cryptographic techniques that are applied to biometric recognition.

(1) *One-way hashing:* One-way hash functions, as discussed in Section 2.1 can be used to convert a biometric input into a message digest. One of the properties of hash functions is the Avalanche effect which states that a small change in the input results in a very different hash output. Biometric inputs from the same user may have slight variations over time. Such variations may lead to a high false rejection rate with a one-way hash function due to the avalanche effect. Thus, one-way hashes are not efficient techniques for biometric recognition.

(2) *Biohashing:* Biohashing consists of extraction and discretization. In the extraction phase, a signal is acquired and preprocessed, and relevant features are extracted as in general biometric recognition systems. The discretization phase, unique to biohashing, consists of four steps. First, a set of pseudo-random vectors are generated using the input and a seed, which are converted to orthonormal vectors. A dot product is applied to the feature vector from the extraction phase and the orthonormal vector. A biocode is obtained using a threshold on the dot product. Any two biocodes can be compared by applying hamming distance [69, 184, 201, 223].

(3) *Fuzzy extractor:* This technique converts biometric data into usable random strings, along with a noise tolerance level. The noise tolerance allows a biometric input to be accepted even if the entered biometric data differs from the stored biometric data by this tolerance distance as both results in the same string being generated [124]. A fuzzy extractor has the following two important functions:

- *Gen*: It is a "probabilistic generation function" that takes user's biometrics Bio_U as input string and produces an l_b-bit extracted string, say $\sigma_i \in \{0,1\}^{l_b}$ as the "user biometric secret key" and τ_i as an "auxiliary string (also called a reproduction public parameter)", that is, $Gen(Bio_U) = (\sigma_U, \tau_U)$.
- *Rep*: This is a "deterministic function" that recovers the "original biometric secret key σ_U" with the help of the helper data, i.e., public reproduction parameter τ_U and biometrics input, say Bio'_U, that is, $Rep(Bio'_U, \tau_U) = \sigma_U$ provided that the "Hamming distance between Bio_U and Bio'_U is less or equal to a pre-defined error-tolerance threshold value *et*".

2.4 Blockchain technology

In a network of systems that share vital information and use it for further processing, usually, a central authority is assigned the task of maintaining the correctness and validity of the information. When such a centralized system is prone to failure, it can lead to a massive loss of vital data needed in critical applications. To overcome such disastrous problems, it is convenient to move the authoritative tasks to a decentralized system.

Blockchain technology is a decentralized system that allows multiple systems to maintain a local copy of a public ledger, called a blockchain, by executing a commitment protocol to add a new block into the blockchain with a process called *mining*, and a *consensus protocol* to ensure consistency among various local copies [156]. Such a system avoids single point failure by allowing multiple authority points with multiple nodes such that if one node fails, all the nodes connected to it will be redirected to another node as long as there is no disconnection of the network. This can be further improvised as a distributed system with no central nodes, where all the nodes cooperatively maintain the vital data [241]. A blockchain is a data structure with a number of data blocks linked linearly in chronological order to form a chain and protected using cryptographic techniques. A block consists of a header with the hash of the previous block, the root of the Merkle hash tree, and the time in seconds when the block was added [83]. It is worth noticing that the contents of a block may vary from one application to another application [245, 246].

2.4.1 Blockchain features

In the following, we list the core features of blockchain as follows [354]:

- *Decentralization*: In a decentralized system, there are several trusted agencies that cooperate collectively to maintain the data in a blockchain [185].

- *Persistence*: The data in a blockchain is tamper-resistant as any given data is validated by multiple nodes and replicated in the local copies of all the nodes. Any change in one of the copies can be clearly identified when the copies of the ledger differ in their content. Blockchains are also made tamper-proof by disallowing the deletion of blocks.

- *Anonymity*: The blockchain technology allows each node to have multiple pseudonyms/addresses to ensure privacy preservation on the transactions [307].

- *Auditability*: To verify any transaction, a user may access any node in the network and trace the transactions. This feature is possible due to the validation of the transactions before recording them during the mining process. Allowing such verification by a user ensures traceability, and hence, non-repudiation of the transactions by the nodes is also achieved [283].

2.4.2 Blockchain challenges

Zheng *et al.* [354] studies the working of the blockchain system with emphasis on the types of blockchains, their applications in various domains, and directions for research in blockchains. The challenges faced in any blockchain system are as follows [354]:

- *Scalability*: As blocks can only be added to a blockchain and never deleted, the amount of storage space is a primary issue. Several solutions have been studied, including limiting the number of transactions processed per unit time, the trade-off between large and small block sizes, lightweight clients, and partitioned blocks [101, 187, 193, 225].

- *Privacy leakage*: The content of transactions is publicly available, and hence, the blockchain cannot ensure transactional privacy [202, 234]. Also, various techniques are developed to link the users with their pseudonyms as discussed in [79, 203, 231, 284].

- *Selfish mining*: Blockchains are susceptible to collusion from nodes if 51% of the nodes are dishonest. This is also known as a 51% attack. Such nodes could add blocks to the chain that reverse an existing transaction that had been previously validated [64, 280].

2.4.3 Blockchain classification

A blockchain system is a distributed P2P network with three possible roles for a node: a) lightweight nodes, b) full nodes, and c) consensus nodes. A lightweight node stores only the block header. A full node stores a complete replica of the blockchain and is allowed to verify the blocks. On the other side, a consensus node can participate in the mining and consensus process [329]. The type of blockchain appropriate for an application is classified by how the participation of the nodes is allowed in the mining process, consensus process, accessing the network, and the level of decentralization [354]. There are three types of blockchains, which are listed below:

- *Public/Permissionless/Open-Access Blockchain*: A public blockchain allows any node to participate in the reading and adding of transactions, with fees. There is no permission required for a node to participate in the consensus validation of blocks. As any node is allowed, it is not possible for every node to know the identity of every other node in the network. A random topology of the network may be used, resulting in the process of propagating validated transactions usually carried out with the help of hierarchical flooding. The nodes perform peer discovery through a query-response cycle on a fixed set of Domain Name System (DNS) servers and publish their peer lists. To restrict malicious nodes from publishing false peer lists, a reputation system based on penalty score is maintained [329]. A public blockchain is completely decentralized in nature. The blockchain is completely immutable as new blocks of transactions may be stored in varied nodes of the distributed system. It is transparent as the amount of influence that a node can have is directly proportional to the amount of resources it can employ [137]. In such a case, the transaction throughput is very low.

- *Private/Permissioned/Closed-Access Blockchain*: A private blockchain is a blockchain system consisting of pre-determined nodes allowed to access the network to read and add transactions [308]. Only the nodes which are granted permission are allowed to participate in the consensus process. Peer discovery is either lightweight or non-existent, as every node in the network is aware of the identity of the other participating nodes. Propagation of transaction data is very efficient due to limited participants. The blockchain may be tampered with if the majority of the nodes agree [81]. The final consensus can be fully controlled by one organization, and the validated blockchain may also be reversed by appending new blocks. It is owned by a single organization and has a high transaction throughput.

- *Hybrid/Consortium/Shared-Permissioned Blockchain*: A consortium blockchain employs a partially decentralized system which is a hybrid of public and private blockchains in order to support scalability over closed blockchains [77]. Only a permitted set of participant nodes may be allowed to perform core activities such as mining, consensus, and propagation, without any fees. The visibility of the blockchain may be either restricted to the approved consensus nodes or to those nodes that have been authorized for certain access rights, if permissioned, or maybe public. The blockchain may be owned by multiple organizations and has a high transaction throughput. The blockchain may be tampered with if the majority of the nodes agree.

A transaction in a blockchain is a digital exchange of assets. This exchange can be executed according to the transactional models [105, 350]:

- *Unspent Transaction Outputs (UTXO) model*: In this model, every user maintains instances of digital assets received but not yet sent as rows of asset type and its quantities. A transaction is the sum of separate quantities of different assets being spent. Transactions are recorded by deleting a set of rows (UTXOs) from the sender and adding a set of rows (UTXOs) from the receiver in the database. This model provides privacy, scalability, and security, but it complicates parallel execution of transactions as the order is required to ensure the correctness of the database [93].

- *Account-based online transactional model*: It consists of executable bytecode programs, called *smart contracts* that are replicated locally in every node inside the blockchain, which activates appropriate logical computations on all honest nodes simultaneously to maintain consistency. A smart contract is triggered by addressing a transaction to it. This model developed by Szabo [304] is simple and space-efficient.

2.4.4 Consensus algorithms

A comprehensive and detailed study of the existing consensus algorithms has been done by Wang *et al.* [329], Aggarwal *et al.* [45], Nguyen *et al.* [248] and Zhang *et al.* [350]. Some of the most used consensus algorithms are summarized below.

- *Ripple*: Ripple is a client-server based consensus algorithm developed by Schwartz *et al.* [281] that executes in rounds. Every server has a list of servers called the "Unique Node List (UNL)" from which it can accept transaction proposals. Before each round, the server collects its received transactions into a candidate set and publishes it. It

receives the candidate sets from the servers in its UNL and validates them by voting "Yes" or "No". If a transaction receives less than a threshold amount of votes, it is discarded before the next round. In the final round, only the transactions which received more than 80% "Yes" votes are included in the ledger before closing it.

- *Practical Byzantine Fault Tolerance (PBFT)*: An unreliable distributed system consists of both faulty and non-faulty systems. The non-faulty systems must agree on a task by sending messages regardless of the unpredictable behavior of the faulty systems of sending conflicting messages. Lamport *et al.* [211] identified this problem as the "Byzantine Generals Problem" concluding that unanimity can be reached in such an untrusted distributed environment if the number of non-faulty systems is more than three times the number of faulty systems, and not more than half of the network connectivity is faulty. A number of algorithmic solutions are provided in such cases. Castro and Liskov [95] proposed a protocol where one of the nodes is elected as the primary node, and the rest are secondary nodes. A consensus is achieved when $f + 1$ nodes agree that the block is valid, with f faulty nodes in an asynchronous environment.

- *Proof of Work (PoW)*: This consensus scheme requires the *miners* to solve a puzzle, and the block created by the first node that solves this puzzle will be added to the blockchain. The miners compute a hash value for the dynamically altering value of the block header such that it cannot exceed a certain threshold [82]. To arrive at this value, the nodes have to try multiple values for the nonce in brute force. The first node to arrive at such a value is the *winner* and its block is accepted to be replicated in the local copies of blockchains of all other nodes. As more nodes join the role of miners, the processing time, cost, and energy expended increase, thereby reducing the efficiency of this algorithm [245].

- *Proof of Stake (PoS)*: This algorithm introduces penalties in addition to rewards during consensus. A miner deposits a part of his currency as a stake to participate in consensus. If the miner is successful in validating a block, the stake is increased; otherwise, the stake is lost. Only the miners who are rich in currency and are willing to place bets on the block can participate. The identities of the miners who have deposited stakes are known, and a miner who stakes the highest is chosen once its ownership is proved using digital signatures. A rich miner is less likely to commit fraud as they may lose their stake. PoS is advantageous in consuming less energy, less cost and processing fast as compared to those for PoW [195].

Table 2.1: Blockchain consensus mechanisms and their applications

Consensus mechanism	Concept	Resource	Applications
Ripple	Voting in multiple rounds	No resource	XRP ledger [13]
PBFT	Voting	No resource	Tendermint [210]
PoW	Hashing	Computations	Bitcoin [245] Ethereum [339]
PoS	Digital signatures	Currency	PeerCoin [195] SnowWhite [72] Ouroboros [192]
DPoS	Voting	Currency	BitShared Ark EOS
PoB	Address suspension	Currency	SlimCoin [12]
PoL	Random number generation	Intel SGX	Luckychain
PoA	Hashing digital signatures	Computations currency	Decred
PoET	Random number generation	Intel SGX	Sawtooth Lake
PoSp	Maximum plots on disk	Storage space	Storj [170]

- *Proof of Activity (PoA)*: Bentov *et al.* [71] proposed the PoA consensus algorithm where the miners begin with the PoW consensus expending their computational resources until a new block is added is found. Next, it shifts to execute the PoS algorithm to add the miner's rewards to the winner block containing the block header.

- *Delegated Proof of Stake (DPoS)*: In DPoS, the nodes in the network execute a voting process to elect a few third-party witnesses which perform the consensus on behalf of the network. If any of the witnesses misbehaves, they are immediately replaced. The voting influence of a node is directly proportional to the amount of currency it holds. DPoS is a robust protocol that works even if the majority of the witness nodes fail [2].

- *Proof of Burn (PoB)*: Stewart [301] developed the PoB consensus where the miners burn their currency at stake by sending the coins to eater addresses and making them unusable to show their commitment to mining. A node that burns more coins has more

power. It can be verified easily, but the burnt coins cannot be revoked.

- *Proof of Elapsed Time (PoET)*: In this consensus protocol, the miners are made to wait for a random amount of time. The miner whose wait time is completed first becomes the one whose block will be added to the chain. PoET was developed by Intel on a special instruction set called "Software Guard Extensions (SGX)" that ensures trusted code runs correctly in a controlled environment [11]. The amount of time to wait is generated using SGX.

Table 2.2: Attacks on consensus mechanisms

Attack	Affected consensus protocols	Description
Double spending	Most protocols	Repeated usage of token
Selfish mining	PoW	Gain profits by generating blocks privately in a mining pool
Nothing at stake	PoS	Blocks added to all branches in a fork
Bribe attack	PoS	Honest nodes are given incentive to add blocks on private fork
Stake bleeding attack	PoS	Broadcast transactions copied from main chain onto private fork to earn extra fees and increase stake
Fake stake attack	PoS	Increase the smaller valued stakes to higher valued stakes

- *Proof of Luck (PoL)*: Milutinovic *et al.* [237] proposed a simplistic consensus algorithm called Proof of Luck (PoL), where the miners generate random numbers, and the block created by the miner with the highest generated random number is added to the blockchain. It is worth noticing that PoL requires all the nodes to be time-synchronized to generate random numbers simultaneously.

- *Proof of Space (PoSp)*: Dziembowski *et al.* [129] proposed a consensus algorithm in which the miners would expend some unit of disk space instead of computational effort. The miners in PoSp generate numerous plots on the hard disk. The miner with the largest number of plots wins.

In addition, Wang *et al.* [329], and Tschorsch and Scheuermann [315] described an abstraction process for development of Proof of Concepts (PoX) consensus algorithm and study the PoX-related schemes in detail. Table 2.1 gives a summary of the discussed consensus mechanisms. Finally, we provide a summary of possible attacks on the consensus algorithms in blockchains based on Zhang and Lee's analysis [351], which is shown in Table 2.2.

2.5 Summary

In this chapter, we have discussed the mathematical preliminaries that are necessary to design authentication schemes in this thesis. One-way hashing is discussed in detail, along with the properties and the existing algorithms. We have studied the elliptic curve cryptography starting with the elliptic curve arithmetic, Diffie-Hellman problems along with encryption-decryption algorithms and the ECDSA signature algorithm. We have also discussed the pros and cons of various biometric verification methods and blockchain technology. In the end, we have described the AVISPA tool, hyperledger sawtooth, Node.js, and MIRACL library that are used to analyze the security and efficiency of the designed schemes in detail.

Chapter 3

Literature Survey

3.1 Existing surveys

This section discusses the literature on the state-of-the-art surveys and review work in the field of smart agriculture by Yang *et al.* [344], Zanella *et al.* [267], Ferrag *et al.* [139], Gupta *et al.* [155], Farooq *et al.* [134], Khanna and Kaur [190], Ruan *et al.* [272], Elijah *et al.* [131], Brewster *et al.* [91], and Ray *et al.* [266].

A comparison of these review surveys is presented in Table 3.1 based on the focus of the paper, the number of agriculture-related papers considered for the study of testbeds, and the security protocols. In addition, other studies by the authors in [67, 84, 149, 175] show the existing technologies and security protocols in IoT-based agriculture. The authors in [337] employ literature review, interviews, and survey to understand the agriculture scenario and adoption of security in IoT-based agriculture in industry and academic research in various parts of the United Kingdom. Our survey work presents a more comprehensive review of protocols to achieve authentication in the IoT-based agriculture field along with security protocols in other related areas of smart farming.

3.2 Existing IoT-based agricultural testbeds

This section presents the technical details of some of the testbeds implemented for varied purposes in agricultural environments based on IoT.

PotatoNet is a wireless sensor network system to monitor an outdoor potato field with 9 INGA nodes in the field, each paired with a programming platform on OpenWRT Linux and a central box that manages the nodes using a cellular data card and distributes power using

Table 3.1: Existing literature surveys on IoT-based agriculture

Paper	Year	Key Highlights	Security	Testbeds
Yang *et al.* [344]	2021	* Development modes and technologies in smart agriculture * Security challenges and solutions in smart agriculture	Yes	No
Zanella *et al.* [267]	2020	*Study of farming resource based layer-wise attacks *Concludes that most security solutions are at application layer *Application resource based required improvements analysis	Yes	No
Ferrag *et al.* [139]	2020	*Threat model based classification of attacks *Study of adoption of generalized security protocols into smart farming * Blockchain based security solutions studied * Future research directions	Yes	No
Gupta *et al.* [155]	2020	*Real-world smart farming use-cases * Open research challenges	Yes	No
Farooq *et al.* [134]	2019	*Farming technological component and network solutions *Cloud, Big-data oriented architectures *Study of existing apps *Security and threat model *Farming policies, strategies and industry trends	Yes	No
Khanna and Kaur [190]	2019	*IoT communication protocols *Study of cloud, WSN, ML in agriculture	No	No
Ruan *et al.* [272]	2019	*Framework of green IoT-based agriculture *Issues in technical, finance, operations and management of IoT in agriculture	No	No
Elijah *et al.* [131]	2018	*Agricultural IoT ecosystem Agricultural sensors *Communication technologies in Smart agriculture *Applications of IoT-based agriculture	No	Yes
Brewster *et al.* [91]	2017	*System of systems architecture for IoT in agriculture *Study of IoT-based large scale pilots for agrifood sector	No	No
Ray *et al.* [266]	2017	*Comparison of IoT based hardware platforms, IoT cloud platforms and sensor system *Practical case studies	No	Yes

passive Power over Ethernet (PoE) with a DSL connection to the Internet [204]. PotatoMesh is an extension of PotatoNet with the addition of a new class of nodes that have the capability to perform at two levels of processing and communication. Data from nodes is sent to a central sink node via an intermittently available uplink connection with the cellular network. Communication among the nodes is fulfilled using low-power radio link such as LoRa (Long Range) or IEEE 802.15.5. Each of the nodes is solar powered using 20Wp photovoltaic cells in addition to 12V, 15Ah lead batteries. Amphisbaena controller on a 32-bit ARM microcontroller with DTN implementation on FreeRTOS is used for the high power platform, and Raspberry Pi is used for the low power platform [146]. Hartung *et al.* [162] summarized the learnings from the PotatoNet and PotatoMesh projects in using third party components, the effect of temperature, dust, rain, and animals on the on-field devices, the effect of farming

activities on the devices and problems due to configuration of systems. Such a detailed analysis of these systems' positive and negative experiences allows newer implementations to be careful of such failures.

Chowdhury *et al.* [104] designed an indoor vertical hydroponic system that works automatically independent of the external climate. A hydroponic system allows the cultivation of plants in water with the required nutrients, minerals, and a stable pH without using soil. The testbed uses the nutrient film technique (NFT) for hydroponic plant cultivation, 6K3R4 and K6 LED for artificial lighting in three tiers, Atlas Scientific pH kit and Atlas Scientific Conductivity Kit, YF-S201 Hall Effect water flow sensor send their reading to web server, through a WiFi module ESP8266, connected to a microcontroller to forward the collected sensor data to the opensource IoT platform ThingSpeak [37].

Swain *et al.* [303] implement a simple remote monitoring system in an open agricultural farm with a sensor node, a gateway, and a handheld device that transmit sensor data to a cloud server using the low power long-range LoRa technology [32, 86, 290]. The sensor and gateway are embedded with a 433-MHz LoRa transceiver communication along with an ESP 8266 WiFi module in the gateway, whereas the handheld device consists of both LoRa communication and ESP8266 WiFi module. The data are recorded on a Blynk cloud server and can be monitored on the Blynk dashboard [26]. The performance of the network is characterized using "matrix laboratory (MATLAB)" simulation [33]. Note that MATLAB is considered as "a programming and numeric computing platform used by millions of engineers and scientists to analyze data, develop algorithms, and create models". An experimental study of this testbed in multiple cases shows that as the height of the end node and gateway node is increased, the link budget strength, coverage area, and signal strength increase when all parameters are maintained constant. Also, increasing antenna height and gain can increase the coverage area.

Pujara *et al.* [262] design an agricultural monitoring system called E-sense that uses Rain Sensor-FC 37 to detect rain, LM 393 module to detect light intensity and brightness, MQ 135 and MQ 9 sensors to detect air quality and CO concentration respectively, CD 4051 multiplexer to multiplex all the analog values from the sensors to the single analog pin on ESP 8266 WiFi module. Blynk application [26] is used to display the sensor data in a user-friendly manner and control the hardware over the Internet. ThingSpeak with the built-in MATLAB analytics [37] is used to collect and analyze a large amount of data in the cloud.

Control of lighting in an artificially controlled agricultural greenhouse system targets reaching a specific level of photosynthetic photon flux density (PPFD) by mixing color

ratios of blue, red, or UV light and is conceptually called a light recipe. Jiang *et al.* [180] proposed a testbed that achieves a customizable light recipe with adjustable PPFD output in two stages. The first stage involves using a multi-input multi-output (MIMO) feedback control loop with daylight harvesting monitored with MATLAB/Simulink simulation. The second stage uses two 1Kw halogen lamps that emulate the change in daylight using the local data set of solar PPFD values collected every hour. A Smith predictor compensates for the time delay between signal reading and delivery, achieving closed-loop stability. Raspberry Pi controller with 4G "Long-Term Evolution (LTE)" cellular router act as the control network center. RESTful "Application Programming Interface (API)" based communication module in Python3 controls the "light-emitting diode (LED)" lights.

Martinez *et al.* [229] applied a middleware technology called FIWARE [29] to collect, transmit and process unknown and unplanned large data sets from precision agriculture that is sent to publish/subscribe architecture that discovers interesting relationships and data flows. The FIWARE IoT architecture consists of an IoT services enablement chapter that manages the device resources and a data /context management chapter with a publish/subscribe broker and big data analysis. IEEE 802.15.4 or ZigBee protocol is used for device communication with a protocol adapter to hide the various communication protocols in a heterogeneous network. The cloud service infrastructure consists of Orion context broker [27], MongoDB database [34] and Cygnus-NGSI [28] to manage context information.

Sadowski *et al.* [275] developed a solar-powered monitoring system for IoT-based agricultural environment using Arduino Uno Rev3 microcontroller, Series 2 XBee with 2mW antenna for wireless communication on a ZigBee mesh network capable of connecting hundreds of nodes, Grove soil moisture sensor, DHT22 temperature and humidity sensor with 0.3 degrees of accuracy for temperature and 2% error rate for relative humidity, small-sized Star Solar D165X165 monocrystalline solar panel with an output of 6V at 3.65W that are placed in a two-hop network such that a relay node forwarded data from sensors to destination.

Escolar *et al.* [132] developed an energy-harvesting prototype device under the PLATINO research project that collects data regarding interesting variables from the surrounding environment under varied conditions of energy and weather. It studies the various LP-WAN technologies in detail and uses the LoRa network with n PLATINO end devices placed over the field with a drone collecting data from them twice a day. Communication between the devices and the drone uses a mixed approach of "Time Division Multiple Access (TDMA)" and "Carrier Sense Multiple Access (CSMA)". Marcu *et al.* [228] presented a detailed comparison of available platforms for implementing agricultural environment using IoT.

Table 3.2 summarizes the testbeds studied in this section.

Table 3.2: Existing testbeds on IoT-based agriculture

Testbed	Hardware	Communication	Software
PotatoNet [204]	INGA nodes Power over Ethernet (PoE)	DSL Internet connection	OpenWRT Linux
PotatoMesh [146]	INGA nodes Amphisbaena controller	LoRa or IEEE 802.15.5	DTN on FreeRTOS for low power Raspberry Pi for high power
Chowdhury *et al.* [104]	6K3R4, K6 LED Atlas Scientific pH kit Atlas Scientific Conductivity Kit YF-S201 Hall Effect water flow sensor	ESP8266 WiFi module	ThingSpeak
Swain *et al.* [303]	ATMEGA 328 p board with gas sensor, ultrasonic sensor, soil moisture sensor Raspberry Pi board Arduino Uno board	433 Mhz LoRa ESP8266 WiFi module	MATLAB Blynk
E-sense: Pujara *et al.* [262]	Rain Sensor-Fc 37 LM 393 module -light intensity MQ135 - air quality MQ9 - CO concentration	ESP8266 WiFi module	ThingSpeak MATLAB Blynk
Jiang *et al.* [180]	1kW halogen lamps Raspberry Pi controller	4G LTE cellular router	MATLAB or Simulink Python3
Martinez *et al.* [229]	-	IEEE 802.15.4 or ZigBee	FIWARE middleware MongoDB Cygnus-NGSI
Sadowski *et al.* [275]	Grove Soil sensor DHT22 temperature & humidity sensor Star Solar D165X165 monocrystalline solar panel Arduino Uno Rev3 microcontroller	Series 2 XBee with 2mW antenna	–
Escolar *et al.* [132]	SHT10 Mesh-Protected and Weather-Proof sensor Adafruit Feather M0 microcontroller Watts photovoltaic solar panel (PV) INA219 sensors	RFM95 LoRa radio	–

3.3 Security protocols in smart agricultural systems

This section discusses the security protocols applied in various subsectors of smart agriculture.

3.3.1 Supply chain and food traceability

Agricultural Supply Chain (ASC) is a “farm-to-fork” logical chain of events comprising the stages - production during farming, processing, transport, storage, and distribution of various

types of agricultural products. Salin [278] highlights the unique features of ASC that make food quality control and safety highly dependent on the efficiency of the management of the supply chain that can be improved by incorporating information technology. Proper planning and decision-making are essential for the proper functioning of ASC. Ahumada [48] provides a detailed review of the planning models that can be used for the agri-food supply chain. Lezoche *et al.* [212] studied the digital technologies that can transform the agri-food supply chain system and their functional, economic, environmental, social, business, technological impact along with organizational, social, and technological challenges to be faced in incorporating them. Bosona and Gebresenbet [87] show the need and benefits of food traceability for supply chain management along with the technological advancements in this direction. Hassija *et al.* [164] studied the various security vulnerabilities, critical application areas of security in the supply chain, improvements needed in the modern supply chain, and technologies available for supply chain security.

Ruiz-Garcia *et al.* [273] studied the applications of "Radio-Frequency Identification (RFID)" in agriculture along with the range of frequencies used and the limitations of its use. Costa *et al.* [108] showed how RFID technology has been used in food traceability in agrifood supply chain. Gandino *et al.* [143] presented a framework and related case studies to automate RIFD-based traceability in the agri-food sector. Alfian *et al.* [50] integrated wireless sensor networks for monitoring temperature and humidity and data mining techniques to predict any missing sensor data on an RFID-based traceability system and shows the improved performance and data accuracy of sensor data. Alfian *et al.* [51] improved the efficiency of the RFID-based traceability system for perishable systems proposed above using IoT sensors to collect temperature and humidity and a machine learning model to improve the performance of the RFID gate.

Badia-Melis *et al.* [63] studied the shift in trends for traceability and conceptual advancements such as traceability centralization with a common framework. Dandage *et al.* [111] studied that food safety fraudulence along with arbitrary inconsistencies are very common in Indian food traceability systems as the 2D barcode is the prominent traceability method even though advanced methods are being implemented. Feng *et al.* [136] studied the development and evaluation methods for using blockchain technology to improve sustainability in food traceability along with its benefits and challenges. Wang *et al.* [328] proposed a consortium blockchain and smart contacts-based framework for tracking the workflow, traceability, and shareability of agri-food supply chains that was practically implemented to realize disintermediation and tracing of farming product information using QR codes. Salah *et al.* [277]

gave a case study of using smart contracts for blockchain-based traceability of soybean and recorded improved transparency in the system. Dasaklis *et al.* [119] used smart contracts to determine the optimal granular size of units that can be efficiently tracked. Bhutta *et al.* [78] proposed a computationally and economically efficient framework based on the Internet of Things (IoT) and blockchain that allows stakeholders to automatically update the quality of perishable goods along with prediction models to maintain backorders. Lin *et al.* [217] designed a food supply chain that incorporates food safety mechanisms using a closed-loop supply chain. Zheng *et al.* [353] constructed a food safety traceability system using 2-dimensional RFID code and big data to store information in an IoT network implemented on rice that gives a standardized third-party certificate regarding the food product.

Future research directions in this area include the development of a unified central model that can be used for security developments with IoT, AI, cloud computing, fog computing, and edge computing. Such models should consider a standardization that allows governmental and private business communities to work together.

3.3.2 Cyber-physical searching

Li *et al.* [123] proposed a framework that provides farmers with the convenience of searching for the information required to manage the production activities on an information-centric network (ICN). This framework consists of four layers: the data source layer consisting of various types of sensor nodes that collect information from the surrounding environment; the data aggregation layer that consists of data aggregation nodes (DAN) with an energy harvesting system and a capability to aggregate data from the sensors present in its communication range; data transfer layer consisting of drones that retrieve data from the DANs; cloud control layer that can search and process user requests and queries. All these entities in the different layers are connected via an information-centric network that sends interest messages from an upper layer to the lower and data messages from a lower layer to the upper. In order to preclude attackers from extracting information from the interest and data messages, the naming information usually visible in ICN is hidden as noise by adding fake messages that are tuned adequately so that attackers cannot identify them as noise.

There is much scope for research in cyber-physical searching in the agriculture sector. With machine learning, deep learning [56, 163], big data analysis can provide a multifold view of the data collected from the agricultural networks and help in better structuring and storage of data which enhances the search operation.

3.3.3 Data confidentiality and privacy

Ametepe *et al.* [55] extended AES-128 [8] with checksum creation, data segmentation, and shuffling of data features in order to add privacy to the field sensor data. Vidyashree and Suresha [321] use AES-128, and SHA-256 [232] to provide data confidentiality, integrity and authentication to sensor data collected from a wireless sensor network based agricultural system. Techniques such as attribute-based encryption schemes, identity-based encryption, and broadcast encryption can be explored for confidentiality and privacy management in agricultural networks.

3.3.4 Access control

Chukkapalli *et al.* [106] proposed a smart farm ecosystem with three modules with ontology for attribute-based access control. The first module presents a three-layered architecture for smart farming: physical entity layer, digital twin layer, and interactions layer. The physical entity layer consists of Farm Based Unit (FBU) with immovable devices like sensors placed in the field, On Board Unit (OBU) with movable devices in the field, Worker Based Unit (WBU), which represents the human resources on the field and Home Based Unit (HBU) which connects all the units to the cloud via a gateway hub. The digital twin module consists of the virtual representations of all the physical entities to monitor the data flow between them. The interactions module defines all the possible interactions between the physical entities and stores them in a representation graph. Based on this architecture, several use-cases for read, access, and operate permissions are defined for access control that is dynamically decided based on the entity's attributes.

Future directions in research of access control include developing agricultural software networks that apply mandatory access control, discretionary access control, rule-based access control, and role-based access control appropriately at various levels to produce and assign permissions.

3.3.5 Data management/data aggregation

A data management system for IoT-based agricultural system with an integrity monitoring system using blockchain, fog computing, and software-defined network is proposed by Friha *et al.* [141]. A data management algorithm is designed that creates a key-value pair for the data from sensor devices in a field. If the value exceeds a threshold, a data array including the current timestamp is sent to a blockchain client. The blockchain client is a fog node

that is responsible for formatting the received sensor data, creating a transaction, signing the transaction using ECDSA [186], assembling a batch of transactions with a batch header and batch signature with ECDSA to form a block on the blockchain, and transmit a packet consisting of the newly created block, along with the validator node socket address to an SDN enabled virtual switch. If no matching flow label exists for the received packet, the switch forwards the packet to the SDN controller that either adds an entry to the flow table or routes the packet based on the existing flow entry in the flow table. This makes sure that there are no errors in the delivery of controls or information, thus monitoring integrity in the system. The performance is evaluated after implementation on Hyperledger Sawtooth and launching a DDoS attack that shows the attack is stopped before the number of blocks escalates.

Song *et al.* [293] proposed a privacy-preserving data aggregation scheme for smart farming with a simple architecture of user with a smart device, cloud, and a control center with two major phases: data collection phase where the farmer's data is uploaded to the cloud after encryption with ElGamal scheme, signed using ElGamal signature scheme and a message authentication code which are verified by the cloud before storage. In the aggregation phase, first, an aggregation space is selected by the control center, and the cloud computes the sum of all ciphertexts in that space. This aggregated ciphertext sum, along with the authentication code, is sent to the users whose data are in that aggregation space. The user verifies the integrity and freshness of the received message before accepting to participate in aggregation. If the aggregated ciphertext result is verified, the user computes a decryption piece, and an authentication code is sent to the cloud. The cloud then computes the aggregated plaintext sum using the decryption pieces received from the different users.

Yousefi *et al.* [346] reviewed various techniques for data aggregation in different fields on Internet of Things (IoT). Zhou *et al.* [356] designed a privacy-preserving algorithm that encrypts data using BGN homomorphic encryption, compares and updates encrypted data on a fog node on an agricultural environment. The sensor data is first stored using K^2-Treap that is efficient for range-max query and dynamic update. Karthickraja *et al.* [188], Ahmed and Biradar [46], [47], Kim *et al.* [194], Sankar *et al.* [279], Yuan *et al.* [347] proposed various methods for data aggregation in smart farming without incorporating any security measures. Stamatescu *et al.* [300] proposed an approach for processing and analysis of data that is hierarchically aggregated during distributed monitoring of crops, without any focus on security.

Data aggregation and management systems that will be developed in the future should

focus on providing scalability, interoperability, and adaption to uncertain dynamic factors of the agricultural surroundings. They should be involved in the entire lifecycle of Agriculture 4.0.

3.4 Non-blockchain based authentication schemes

This section studies various authentication schemes proposed in smart agriculture without any involvement of blockchain technology.

Ali *et al.* [52] proposed an authentication scheme that allows a user to access real-time environment data from sensor nodes in a wireless sensor-based remote agricultural monitoring network. The sensor data is passed from the sensor nodes to the gateway via an access point, which is then forwarded to the base station. The user is allowed to access the data after registration and verification by the base station. During registration, a user enters biometric data, which is processed via the biometric generation function of the fuzzy extractor, and receives a smart card from the base station consisting of secure verifiable parameters and the biometric public parameter. The user logs in with identity and password credentials along with the smart card. The biometric reproduction function gives the secret biometric fuzzy parameter, which is then used to verify the secret smart card parameters. Once verified, hash functions, XOR operations, and symmetric encryption/decryption are used to authenticate the user and establish a session key. Though their scheme is lightweight, it suffers from several security pitfalls, such as privileged-insider and "Ephemeral Secret Leakage (ESL)" attacks, and it fails to support anonymity and untraceability issues.

Chen *et al.* [102] improved the above scheme by making the static security parameters in the smart card dynamic to remove traceability. It ensures that a shared secret that is part of the session key and shared with all participants is restricted to be available only to the base station and sensor node, overcoming user impersonation attacks and adding perfect forward secrecy and user anonymity. A denial of service (DoS) attack is avoided by sending the updated user parameters in the password update phase to the base station for processing. Further, the high computational and communicational cost is reduced by replacing symmetric encryption/ decryption operations with hash functions.

Chae and Cho [96] proposed an authentication scheme for a P2P greenhouse smart farm with some IoT sensing and actuator devices placed inside the greenhouse and some IoT sensing and actuator devices placed outside. A user wanting to access the devices needs to be authenticated before access is provided. Similarly, any two devices should be authenticated

before any form of communication commences. The external device advertises its signature and certificate. The user sends a message with the authentication request, IP addresses and random nonces of the external and internal devices, user certificate, and digital signature to the external device, which is forwarded to the internal device. Once the user is verified, the internal device sends its certificate, digital signature, IP addresses of all three entities, and nonces to the external device. After verifying the signature and certificate, the external device forwards the same to the user. The user sends its digital signature to the internal device, which verifies the user's public key with an authentication authority and uses it to verify the digital signature. Once verified, the user and internal smart device share a session key. This scheme has a very high cost as it utilizes public-key cryptography for digital signatures and certificates. It also does not support user anonymity, and device anonymity as their IP addresses are shared in messages publicly. It is also vulnerable to MiTM, replay, physical device capture, and user and device impersonation attacks. In addition, the scheme does not define the association between internal and external devices.

Table 3.3: Computational costs comparison of non-blockchain authentication schemes

Scheme	Operational cost	Cost in ms
Ali *et al.* [52]	@ smart device: $11T_h + T_{fe} + 3T_{senc}/T_{sdec}$ @server: $8T_h + 5T_{senc}/T_{sdec}$	$\approx$ 5.738 ms $\approx$ 0.445 ms
Chen *et al.* [102]	@ user & smart device: 20 T_h @ server: 17 T_h	$\approx$ 6.18 ms $\approx$ 0.935 ms
Chae and Cho [96]	8 T_{ecm} + 8 T_h + 2$Teca$	$\approx$ 20.808 ms
Rangwani *et al.* [265]	@user/sensor: 8 T_h + 5 T_{ecm} @gateway: 7 T_h + T_{ecm}	$\approx$ 13.912 ms $\approx$ 1.059 ms

Bothe *et al.* [88] focused on the sovereignty of the data collected in a smart agricultural system. They study the different possible mechanisms to secure the communication channels that are established in an IoT-based smart agricultural system. They also proposed a generalized security architecture based on the ODiL framework for farm information management systems. An authentication scheme is implemented on the agricultural machine using the ODiL platform with the OAuth framework that uses RFID to continuously poll users' credentials and achieves continuous user authentication.

Rangwani *et al.* [265] proposed a two-factor remote user authentication scheme for an agricultural wireless sensor network such that a gateway node allows the sensor nodes in

the field to interact with the remote user, cell phone user, and the database server present outside the field environment via the internet. The architecture is intended for the purpose of a regular remote surveillance of the field. The scheme uses elliptic curve cryptography, hashing operation, and symmetric encryption/decryption. This scheme uses only two factors - user credentials and user biometrics for the authentication of the user.

3.5 Comparative analysis of non-blockchain based authentication schemes

Table 3.4 provides a comparative analysis in terms of the communication cost, Table 3.3 gives computation cost based on the experimental values of cryptographic operation obtained using MIRACLE library [38] in Table B.3. Table 3.5 presents a summary of the schemes discussed. Table 3.6 lists the pros and cons of each of the authentication schemes in the agricultural domain. The analysis shows that very few protocols have been developed for authentication and these protocols have security drawbacks that need to be fixed. This shows that authentication in smart farming has a lot of scope for further exploration and research. There is a need to develop more wholesome authentication protocols.

Table 3.4: Communication costs comparison of non-blockchain authentication schemes

Scheme	No. of messages	Cost in bits
Ali *et al.* [52]	5	5504
Chen *et al.* [102]	4	4960
Chae and Cho [96]	4	12896
Rangwani *et al.* [265]	5	4128

3.6 Blockchain-based authentication schemes

In this section, we review some of the competing existing security protocols, including blockchain-based solutions and authentication protocols proposed in the literature related to IoT-enabled agriculture and other related environments. In recent years, authentication

Table 3.5: Concept summary of non-blockchain authentication protocols in smart agriculture

Scheme	Cryptographic Concept	Attacks Resistant To
Ali *et al.* [52]	*Hash functions *Symmetric Cryptography	* Replay attack * Malicious device deployment attack * Smart device/drone capture attack
Chen *et al.* [102]	* Hash functions	* DDoS attack * Privileged insider attack * ESL attack
Chae and Cho [96]	* Digital Signature * Digital Certificate	* Brute force attack * Dictionary attack
Rangwani *et al.* [265]	* Elliptic Curve Cryptography * Symmetric Cryptography	* Man-in-The-Middle attack * Replay attack * Impersonation attack * Stolen smart device attack

Table 3.6: Advantages and drawbacks of non-blockchain authentication protocols in smart agriculture

Scheme	Advantages	Drawbacks
Ali *et al.* [52]	* Comparable storage cost * Low computational cost	* Vulnerable to privileged insider attack, DoS attack, ESL attack * Does not support anonymity, untraceability
Chen *et al.* [102]	* Low communicational cost	* Vulnerable to physical device capture and stolen smart card attack
Chae and Cho [88]	* Reduces performance degradation of smart palm devices * Handoff service of mobile device considered for execution time	* No user/ device anonymity * No traceability * Vulnerable to physical device capture impersonation, replay, and MiTM attacks
Rangwani *et al.* [265]	* Supports anonymity, untraceability & perfect forward secrecy	* Only two factors for authentication

became one of the potential security services in various networking environments to secure the networks [99, 113, 114, 116, 127, 214, 215, 216, 239, 250, 331, 332, 334].

3.6.1 Public blockchain-based schemes

Almadhoun *et al.* [53] proposed a system with IoT devices which are assigned to a fog node. The fog nodes have the capability to perform computations required during the authentication process on behalf of IoT devices, thus reducing their burden of processing.

A smart contract contains the association of the fog nodes to its linked IoT devices [206]. The administration is responsible for managing the permissions of which user is allowed to access which device and perform what operations. The cloud consists of huge storage and computation servers that store the enormous data collected by the IoT devices. The end-users require permission to access their IoT devices. All the entities except the IoT devices have unique Ethereum addresses and interact with the smart contract directly or through an application. Their scheme supports confidentiality, integrity, and non-repudiation, and it is also resilient against Denial of Service (DoS) attacks [288].

The Public Key Infrastructure (PKI) is used to obtain the public key of an entity. The PKI is usually realized as an organized tree with a root. This leads to a centralized structure where the data of all the other entities is vulnerably dependent on the root. To overcome this issue, a decentralized blockchain-based PKI has been proposed that considers thin clients to perform the same functions as a full node user. In a blockchain based-PKI, an entity creates two key pairs: a) online private-public key pair and b) offline private-public key pair, and adds the details onto the blockchain along with the signatures after being verified by the miners. To obtain a key corresponding to an identity, the most recent block associated with that identity is retrieved, which has either the key or an update that the key that has been revoked [323].

Jiang *et al.* [182, 183] in their works had come up with the following innovative schemes for authentication. In their first scheme, called "Privacy-preserving Thin-client Scheme (PTS)" [182], Alice, being the initiator, first sends her identity and public key to Bob, the responder. Bob authenticates Alice by sending her identity along with $k-1$ random identities to the full node users. Each of the full node users traverses their blockchain to retrieve the public keys corresponding to the identities received and sends them to Bob. If the public key of Alice's identity received from all the full node users is the same, Alice's public key is verified as true. Then, Bob sends his identity and a nonce encrypted with Alice's public key to Alice. Alice decrypts Bob's identity and nonce with her secret key, retrieves Bob's public key from her blockchain, and verifies Bob's pubic key with the full node users. Then Alice sends her nonce, Bob's nonce, and her identity encrypted with Bob's public key. After decrypting the message with his secret key, Bob verifies the correctness of his nonce, encrypts Alice's nonce with Alice's public key, and sends it back to Alice. Once Alice verifies the received nonce as true, Alice and Bob are mutually authenticated. To reduce cost, they proposed another scheme, called "Efficient Privacy-preserving Thin-client Scheme (EPTS)". It was shown that EPTS could significantly reduce the computational cost of the full node

users and communication cost when k is particularly large.

Jiang *et al.* [183] also proposed a "Privacy-preserving Thin-client Authentication Scheme (PTAS)" which employs the idea of private information retrieval (PIR). To enhance the security of PTAS, they also proposed an $(m-1)$-private-PTAS, which removes the restriction that the number of full node users, say m, should be a power of 2 and that the full node users do not collude. In fact, this enhanced scheme ensures that even if $m-1$ nodes collude together, the thin client's privacy is still preserved. The number of full node searches in PTS is $m.k$, whereas in PTAS it is independent of the number of full node users m and in $(m-1)$-private PTAS, the total searches is $m.k/2$.

3.6.2 Private blockchain-based schemes

Wu and Tsai [340] proposed a system that provides information privacy, integrity, preservation, and accuracy with a rapid authentication scheme. It amalgamates the concepts of blockchain, dark web, bilinear pairings, "keyed-hash message authentication code or hash-based message authentication code (HMAC)", symmetric encryption, 4G mobile communication, and Global Positioning System (GPS) location sensing to deflect "Distributed Denial-of-Service (DDoS) attack". The dark web was applied to create a private blockchain and used for identity authentication before a user is allowed to search. The packets received from the equipment are distributed over the blockchain. A user sends encrypted search data to the trusted authority (TA), which is then forwarded to the dark net servers whose locations are unknown to the user to ensure protection against cyber attacks on them. The blockchain stores messages received from the TA in a block. To ensure the integrity of any message, it will calculate its HMAC and compare it with the HMAC of the stored message. The origin of the message is also verified using the bilinear pairing operations.

3.6.3 Consortium blockchain-based schemes

Cui *et al.* [110] proposed a hierarchical architecture in a multi wireless sensor network (WSN) environment with a hybrid blockchain model. It consists of a public blockchain with the base stations and end-users as the miners to authenticate and register the cluster heads with the help of smart contracts. A local blockchain with the cluster heads as the miners is used to register and authenticate the ordinary nodes using smart contracts. The Ethernet address associated with each node is hashed to obtain its identity, which is further marked as OrdinaryID (OID), ClusterID (CID), and StationID (SID) to designate their roles

vividly. During initialization, the base station generates the public-private key pairs and the IDCard for each node. The cluster head sends a registration request to the public blockchain, which stores its details and allows access to the local blockchain. When the ordinary node sends a request for registration to the cluster head, it verifies the timestamp and triggers the smart contract on the local blockchain to verify its details [157]. Once verified, the node details are stored on the local blockchain and allow access to the cluster network. When the node sends a request to communicate with another ordinary node through the cluster head, it finds the base station of the related WSN that triggers the smart contract on the public blockchain to check if both the requesting and requested nodes exist and have valid IDs and are alive. If both the nodes are in the same cluster, a secure channel is established. Otherwise, the corresponding cluster head nodes exchange the authentication credentials with the transaction voucher obtained from the local blockchain. If the nodes belong to different WSNs, the corresponding cluster head nodes exchange the authentication credentials with the transaction voucher obtained from the public blockchain. Then a secure communication channel is established between the two nodes for further communication. When an end-user wants to communicate with a node, the user sends a request to the base station, which triggers a smart contract to authenticate the user's identity. If it is successful, the credentials are sent to the user and the node to establish a secure connection. Their scheme is resistant to Sybil attack, DoS attack, Man-in-the-Middle (MiTM) attack, as well as a replay attack. Furthermore, it supports scalability, decentralization, and cross-domain authentication.

Yao *et al.* [345] proposed a scheme for authentication in a Vehicular Ad Hoc Networks (VANETs) with Fog computing [209]. It allows computation to be conducted in units closer to the vehicular units based on blockchains. Their system framework consists of an Audit Department (AD), On-Board Unit (OBU), Road Side Unit (RSU), Service Manager (SM), Witness Peer (WP), and a consortium blockchain. The vehicle OBU sends its public key and identity to the AD, which returns part of the public and private key and is used by OBU to construct the complete public and private key, thus successfully performing the registration of the OBU. The same process is used to register SM with AD. When an OBU requests access to RSU, it creates a ciphertext from the IDs of all the SMs and the Lagrange difference polynomial and sends it to RSU. This ciphertext is further forwarded from RSU to the SM, which extracts the OBUs identity and signature and verifies it [208]. Once verified successfully, it searches and updates its local database and forwards the authentication details to the Witness Peer (WP). The WPs store the details in their memory and create a block

after a time period after executing the PBFT consensus algorithm. The scheme ensures confidentiality through encryption at every phase, integrity by denying access on detection of tampering, anonymity by random choice of pseudonyms, traceability, and anonymity by revoking the public key of the identity of the misbehaving entity, and also non-interactivity.

Kaur *et al.* [189] proposed a different scheme for authentication in a vehicular fog network. In their approach, an OBU of a vehicle sends an encrypted concatenation of the OBU's identity, and timestamp to the AD [76]. The AD verifies the timestamp and checks the availability of the identity on the blockchain. It stores an authenticator, say *auth*, as the bitwise XOR of two parts: the first is the hash of the concatenation of the secret key and identity of OBU and the second part is the hash of the concatenation of the secret key of AD and identity of OBU. It then computes and sends the secret and public keys for OBU, along with the second part of authenticator *auth*, which is saved by OBU. When the OBU requests access to the SM, it constructs and sends to SM: the first, authenticator *auth* and the second, authenticator value $Auth_{OBU}$ generated using *auth*, timestamp, OBU identity, and the public variant of the OBU secret key. Once $Auth_{OBU}$ is verified to be valid, SM also constructs another authenticator $Auth_{SM_j}$ that is verified by OBU. A common session key is then computed using the key derivation function on the authenticator *auth*, timestamps of OBU and SM, and OBU identity. These authentication results are added to the blockchain similar to the consensus process as suggested in [310, 345].

Islam and Shin [173] proposed a healthcare system that uses a UAV to read data from a Body Sensors Hive (BSH) that collects health data (HD) from sensors placed on users [160, 165]. The UAV verifies the authenticity of the received data and then forwards it to the server. The server is responsible for adding data as a block into the blockchain after consensus with the Ground Control Station (GCS) and private cloud. For this to work, the details of the user are initially registered on the blockchain using a smart contract. A smart contract is also used to assign a UAV to a set of users using the UAV's public key. Each entity in the system has a private key based on its "media access control address (MAC address)", timestamp, and a random seed, and a corresponding public key is also generated. A trust token is generated for BSH and UAV during registration that is used during communication with the server. To synchronize the UAV with its BSHs, UAV encrypts and sends a shared key with the public keys of all the BSHs. The BSH decrypts the shared key, and double encrypts its public key with first the trust token and then with the shared key. Their system shows resistance against MiTM attacks, replay attacks, illegal data tampering, and unauthorized data access attacks.

Chen *et al.* [103] discussed an innovative platform, known as AgriTalk, that uses precision farming for the soil cultivation of turmeric. Their system automates the fertilization, pest control, and irrigation processes using an IoT environment with sensors, actuators, and controllers. The factors to be monitored for turmeric cultivation are pH, nutrients, electric conductivity (EC), moisture levels, amount of nitrite, Nitrogen (N), Potassium (K), Phosphorus (P), and light. The amount of NPK required for the available amount of EC is determined by quadratic equations, which are then fed into the first layer of a neural network model. This model uses three neurons as the hidden layer and outputs the productivity from the output layer. This is repeated until the bias converges and the output rhizosphere weight is related to the content of the curcumin in the soil using a linear equation. The actual harvested productivity is then compared to the predicted productivity to obtain the growth rate. Similarly, for pest and fertilizer control, a model was established to relate the environmental factors such as the humidity, temperature, and egg hatching period. The AgriTalk system consists of a SnsrCtrl board with soil and insect sensors, a weather station connected to it by wires, an AgriCtrl microcontroller board that controls all the connected IoT devices, and an AgriTalk server in the cloud [309]. To detect any abnormality in sensor functioning, the values of nearby sensors are compared, and the power consumption is monitored. To correct a hardware failure, the device is replaced. To correct a software failure, the device application is modified by calibration without the need to modify the firmware or sensors' operational range. AgriTalk has a good performance for message delay using 4G [158]. They also noticed that the precision of the regression model could be further increased by including more factors like wind speed and direction to predict the rate of eruption of diseases.

Zhou *et al.* [355] proposed an authentication scheme that used the Identity Based Encryption (IBE) as the PKI for the distribution of keys with the integration of blockchain. In their scheme, a user sends an identity, a random seed, and a signature to the Key Generation Center (KGC) to request a private key. The KGC divides the nodes under its control into three groups: a) supervision nodes, b) production nodes and c) protection nodes. The supervision nodes verify the user identity and sign with its master key. After consensus from the other nodes, a validated block is added to the blockchain, and a partial private key is sent to the user. The protection nodes participate in the block verification consensus and key distribution consensus with the help of the Proof of Vote (PoV) consensus algorithm that uses the supervision nodes' signature to generate a partial private key and sends it to a production node. The production nodes also participate in block verification consensus

and transmit the partial private key to the user. After a fixed number of blocks are added, a supervision node changes its role into a protection node, and a new supervision node is re-elected. The users use the private and public keys pair to mutually authenticate each other. The key escrow problem, where the KGC can produce the user's private key to decrypt messages meant for the user, is circumvented by blinding the partial information in the channel in this scheme.

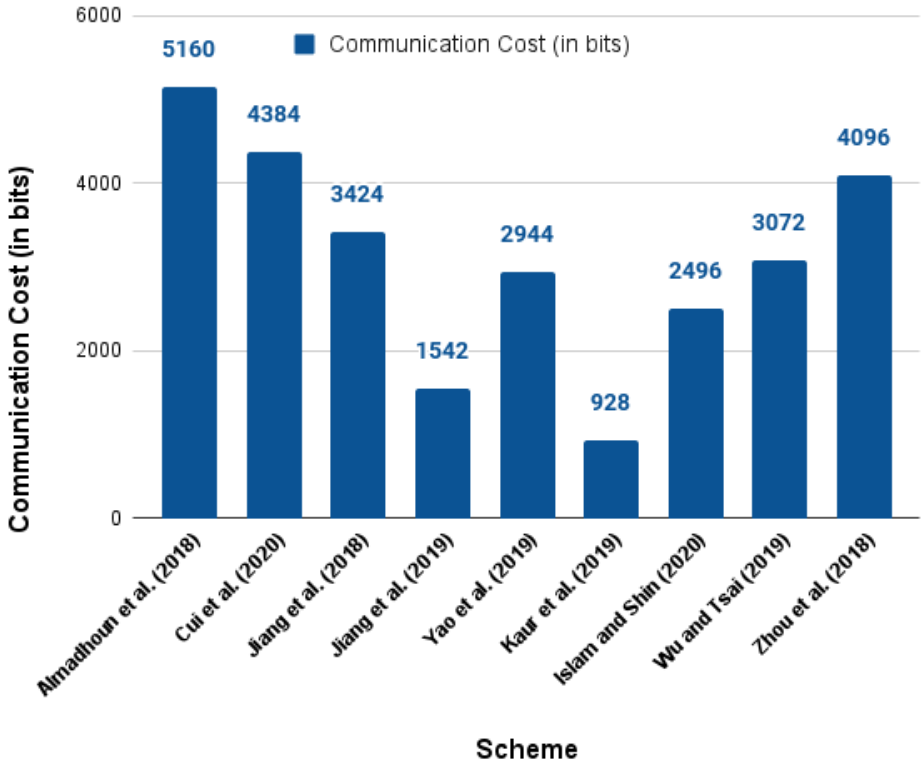

Figure 3.1: Communication costs comparison of blockchain-based authentication schemes

3.7 Comparative analysis of blockchain-based authentication schemes

In this section, we perform a detailed comparative study on communication and computation costs, and also security and functionality features among the considered existing competing schemes, such as the schemes of Almadhoun *et al.* [53], Cui *et al.* [110], Jiang *et al.* [182],

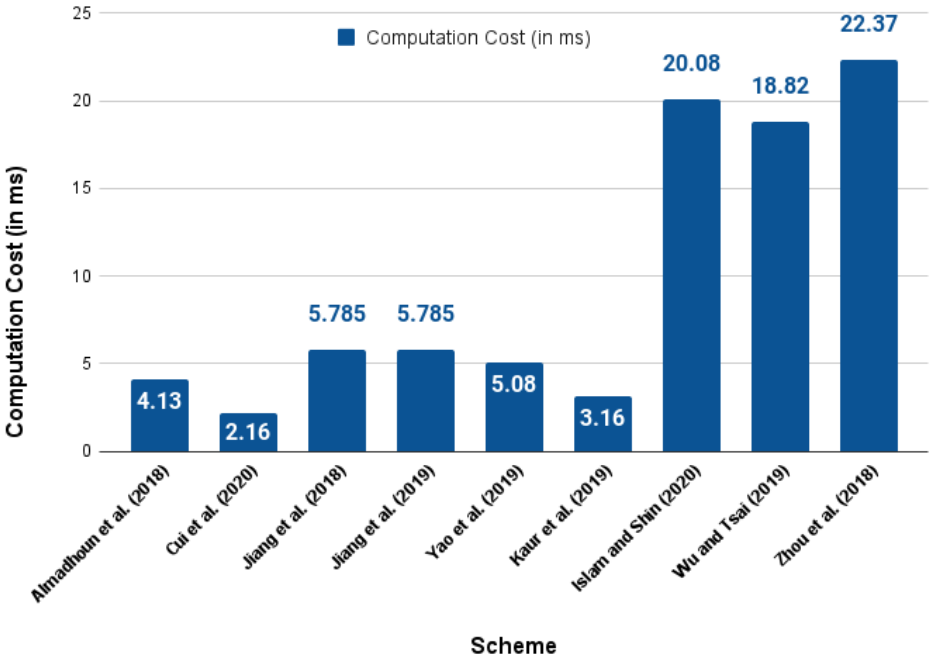

Figure 3.2: Computation costs comparison of blockchain-based authentication schemes

Jiang *et al.* [183], Yao *et al.* [345], Kaur *et al.* [189], Islam and Shin [173], Chen *et al.* [103], Wu and Tsai [340] and Zhou *et al.* [355].

3.7.1 Communication costs comparison

For communication cost analysis, the "identity", "random number (nonce)", "elliptic curve point of the form $P = (P_x, P_y)$ where P_x and P_y are x and y coordinates of P respectively", "hash output (if SHA-256 hash algorithm is applied)", and "timestamp" are 160, 160, (160 + 160) = 320, 256 and 32 bits, respectively. It is also assumed that the security level of an 160-bit elliptic curve cryptography (ECC) is same as that for an 1024-bit RSA public key cryptosystem. Under these assumptions, the comparative study on communication costs among the existing schemes is shown in Figure 3.1. The communication costs needed for online computation for the schemes of Almadhoun *et al.* [53], Cui *et al.* [110], Jiang *et al.* [182], Jiang *et al.* [183], Yao *et al.* [345], Kaur *et al.* [189], Islam and Shin [173], Chen *et al.* [103], Wu and Tsai [340] and Zhou *et al.* [355] are 5160, 4384, 3424, 1542, 2944, 928, 2496, 3072 and 4096 bits, respectively. It is observed that the schemes of Kaur *et al.* [189] and Jiang *et al.* [183] need less communication costs as compared to other compared schemes.

Table 3.7: Security and functionality attributes comparison

Scheme	Security & functionality attributes	Application areas
Almadhoun *et al.* [53]	Confidentiality, Integrity, Non-repudiation	General
Cui *et al.* [110]	Scalability Mutual Authentication, Cross Domain Authentication, Decentralization	Multi-WSN
Jiang *et al.* [182]	Anonymity, Privacy Preserving, Thin Client	PKI-based authentication
Jiang *et al.* [183]	Anonymity, Privacy Preserving, Thin Client	PKI-based authentication
Yao *et al.* [345]	Confidentiality, Integrity, Anonymity, Non-interactivity, Traceability, Non-repudiation	VANETs
Kaur *et al.* [189]	Confidentiality, Integrity, Anonymity, Non-interactivity, Non-repudiation, Mutual Authentication, Key Exchange, Forward Secrecy	VANETs
Islam and Shin [173]	Authentication, Integrity, Authorized Access	Internet of Drones (IoD) in Healthcare
Chen *et al.* [103]	Sensor Calibration Good Message Delay Performance	Smart Agriculture
Wu and Tsai [340]	Identity Authentication, Key Exchange, Location Privacy, Integrity, Information Preservation, Information Accuracy	Smart Agriculture
Zhou *et al.* [355]	Decentralization, Integrity, Mutual Authentication, Key Exchange	PKI-based authentication

3.7.2 Computation costs comparison

For computational costs comparison among the considered existing competing schemes, we consider the experimental results performed in Appendix B where the average execution time for the cryptographic primitives are listed in Table B.3. The comparative study on computational costs among the existing schemes is shown in Figure 3.2. The computation costs needed for exchange of the messages among the entities for the schemes of Almadhoun *et al.* [53], Cui *et al.* [110], Jiang *et al.* [182], Jiang *et al.* [183], Yao *et al.* [345], Kaur *et*

al. [189], Islam and Shin [173], Chen *et al.* [103], Wu and Tsai [340] and Zhou *et al.* [355] are 4.13, 2.16, 5.78, 5.78, 5.08, 3.16, 20.08, 18.82 and 22.37 milliseconds, respectively. It is seen that the computation costs needed in the schemes of Islam and Shin, Wu and Tsai, and Zhou *et al.* are significantly more than other schemes.

Table 3.8: Concept summary of blockchain based schemes

Scheme	Blockchain usage	Cryptographic concept	Resisted attacks
Almadhoun *et al.* [53]	Fog nodes access blockchain and execute smart contracts **Type**: Public	Hash Asymmetric Encryption	MiTM Replay DoS
Cui *et al.* [110]	Local blockchain to authenticate ordinary nodes, Public blockchain to authenticate cluster heads **Type**: Consortium	Hash Signatures	Sybil Attack MiTM Replay DoS
Jiang *et al.* [182]	PKI is stored as blockchain **Type**: Public	Private Information Retrieval	Dishonest Node Collusion
Jiang *et al.* [183]	PKI is stored as blockchain **Type**: Public	Private Information Retrieval	Dishonest Node Collusion
Yao *et al.* [345]	Store a registered vehicle's authentication details to be used for verification **Type**: Consortium	Elliptic Curve Cryptography	MiTM DDoS Impersonation Attack
Kaur *et al.* [189]	Store authenticator & retrieve to verify **Type**: Consortium	Elliptic Curve Cryptography	Replay Impersonation Attack
Islam and Shin [173]	Storage of data after authentication **Type**: Consortium	Symmetric Encryption Signatures Bloom Filters	MiTM Replay Unauthorized Access Data Tampering Attacks
Wu and Tsai [340]	Distribute authentication packets over blockchain **Type**: Private	Symmetric Encryption HMAC Bilinear Pairings	MiTM Data Tampering DDoS Hardware Attacks Server Location Exposure Packet Loss Physical Device Capture Device Location Change Attacks
Zhou *et al.* [355]	Build KGC **Type**: Consortium	Identity Based Encryption	KGC Attack Key Escrow Problem Replay DDoS Session Hijacking

Table 3.9: Advantages and drawbacks comparison of blockchain based authentication schemes

Scheme	Advantages	Drawbacks
Almadhoun *et al.* [53]	Low Computation Cost	1) Does not meet most IoT communication scenarios 2) High Communication Cost
Cui *et al.* [110]	Low Computation Cost	1) No support for Dynamic Node Addition 2) High Communication Cost
Jiang *et al.* [182]	1) Viable for thin clients (smartphone users) 2)Low Computation Cost	High Communication Cost
Jiang *et al.* [183]	1) Viable for thin clients (smartphone users) 2) Need not to download entire blockchain	1) Not resistant to 51% attack
Yao *et al.* [345]	Low Computation Cost	No support for mutual authentication between vehicles and service managers (SMs)
Kaur *et al.* [189]	1) Low Computation Cost 2) Low Communication Cost	ID_{OBU_i} is sent in clear text (no anonymity)
Islam and Shin [173]	1) High Connectivity 2)Low Power Transmission 3) Tested on different hardware	Expensive Computations
Chen *et al.* [103]	1) Curcumin Concentration increased by 5 times 2) 40-60% increased Chlorophyll 3) Software sensor calibration without any modification to the sensor	Less frequent sampling frequency leading to data inaccuracy
Wu and Tsai [340]	1) Lightweight Encryption 2) Multi-faceted security	Expensive Computations
Zhou *et al.* [355]	1) Inexpensive blinding technology for secure channel 2) Role replacement prevents any bias in system 3) Parallel request process enhances speed and scale of the system	1) Expensive Computations 2) High Communication Cost

3.7.3 Security and functionality features comparison

Table 3.7 provides a concise view of the cryptographic attributes supported by each of the existing competing schemes and the domains they are applicable. In Table 3.8, we have summarised the core concept of the schemes by specifying how the technology of blockchain can be leveraged alongside the cryptographic tools used in those schemes. We have also listed various potential attacks that the schemes that are resistant against an adversary who is either passive or active in nature. Furthermore, Figures 3.1 and 3.2 show the costs spent in the computations of various cryptographic operations used and in the communication of messages between various entities involved in the schemes. Finally, Table 3.9 lists the advantages and disadvantages of each studied scheme. It is worth noticing that there is a trade-off among the security and functionality features, core concepts applied, resilience against potential attacks, pros and cons, and also the involved communication and computation overheads in each scheme.

3.8 Compared schemes

This section summarizes the additional schemes whose performance is compared with the proposed authentication schemes in chapter 4 to chapter 7.

Tian *et al.* [311] designed a "privacy-preserving authentication framework" in the Internet of Drones (IoD) deployment that utilizes the lightweight online/offline signature approach. Their scheme does not provide anonymity and untraceability properties and also does not protect against ESL attacks.

Tai *et al.* [305] proposed an authentication mechanism for a heterogeneous IoT-based ad hoc wireless sensor networking environment to withstand the security vulnerabilities found in the existing authentication schemes. However, their scheme still suffers from privileged-insider, malicious IoT device deployment, ESL, and physical device capture attacks. In addition, anonymity and untraceability properties are not preserved in their scheme. It is worth noticing that none of these existing authentication protocols in the IoT-enabled agriculture environment supports the blockchain solution.

Dhillon and Kalra [122] presented the concept of perceptual hashing on the feature vector generated out of the biometric data collected. During their user registration process, the user generates a user-masked password, user-masked identity, and user-masked biometric using perceptual hashing and sends all this information to the gateway node via a secure channel. The gateway creates a gateway-masked user identity by using its secret and gateway-

masked password and biometric by using the shared secret between the gateway and the user. It creates a password entity parameter by mixing a gateway-masked user identity with a gateway-masked password and a biometric entity parameter by mixing a gateway-masked user identity with a gateway-masked biometric. Similarly, the gateway-masked identity and password are generated during the registration of the sensor node. During the login phase, the user-supplied identity and password are used to calculate and verify the password entity parameter and biometric entity parameter. Once these are verified, the login parameter is generated using the password entity parameter, biometric entity parameter, and the shared secret with the gateway. After that, the sensor node identity is selected, and the login gateway masked parameters along with an encrypted nonce are sent to the sensor node. The sensor node sends the encrypted sensor gateway masked password to the gateway. The gateway verifies the sensor and user gateway-masked parameters and then generates user-sensor mixed identity, hidden sensor identity, and encrypted login parameter and sends it to the sensor. The sensor decrypts the user nonce, creates its sensor nonce, generates the common session key using the user and sensor nonce, and also computes the key verifier. The sensor sends its nonce in encrypted format along with the key verifier. The user decrypts the sensor nonce, generates the session key, and verifies it using the key verifier. If it is verified correctly, the user and sensor compute the same session key to be used for further communication. However, this scheme does not support the user device revocation phase and user password/biometric update phase. In addition, it does not protect against user impersonation, stolen smart cards, ESL, and privileged-insider attacks. Furthermore, their scheme is not resilient against DoS attacks because of the usage of perceptual hashing instead of the fuzzy extractor method [98].

Sadhukhan *et al.* [274] proposed an authentication scheme in which the authentication process uses symmetric encryption, and the session key generation uses elliptic curve cryptography (ECC). A user in their scheme has two parameters: a) user digest formed from user credentials and b) a user elliptic curve point. Similarly, the sensor has two components: a) a sensor digest formed from user credentials and b) a sensor elliptic curve point. The user sends its parameters to the sensor after encrypting with a symmetric key shared with the gateway. The sensor encrypts its parameters with its symmetric key shared with the gateway and forwards to the gateway, along with encrypted user parameters. The gateway decrypts the parameters of both entities. It further encrypts user parameters with the symmetric key shared with the sensor and encrypts sensor parameters with the symmetric key shared with the user, and forwards both to the sensor. The sensor verifies that the user is the one it needs

to connect to and computes the session key by multiplying its private integer with the user elliptic point. The user also computes the session key by multiplying its private integer with the sensor elliptic curve point. This scheme has a mutual authentication system and resists replay as well as man-in-the-middle attacks. However, it lacks "user anonymity and traceability properties" and cannot resist "user impersonation" and "ESL and privileged insider" attacks. In addition, it does not support user device revocation, dynamic node addition, user biometric, and password change phases.

Shuai *et al.* [286] proposed another authentication scheme based on the Rabin public-key cryptosystem. During the registration process, the user mobile device stores verification parameters received from the registration authority. During the login phase, it verifies the registration parameters, generates a mixed parameter, and sends encrypted user nonce and a verifier based on the user nonce and mixed parameter to the "industrial management gateway (IMG)". Once the user is verified, the IMG generates its gateway nonce, a shared key using the sensor identity and the Rabin cryptosystem primes. The encrypted gateway nonce and a verifier are sent to the sensor. Once the gateway is verified, the sensor generates its random nonce and encrypts it using the Rabin public-key scheme. It also generates the user shared session key along with its verifier. The encrypted sensor random nonce and the session key verifier are forwarded to the user via the gateway. The user generates the session key verifier using both the user nonce and sensor nonce and verifies it using the key verifier. This scheme supports user anonymity, traceability, "dynamic node addition" and "user password change". Though this scheme resists replay attacks and man-in-the-middle attacks, it cannot resist user impersonation, ESL, and privileged insider attacks. Furthermore, this scheme does not support user device revocation and biometric change phases.

Tian *et al.* [311] developed a scheme for secure communication between Unmanned Aerial Vehicles (UAVs) with the help of a Mobile Edge Computing (MEC) that authenticates the UAVs and MECs leading to the establishment of a session key that can be used for symmetric communication. Each UAV is issued multiple certificates during registration and a private-public key pair during initialization. An offline signature is prepared without private-public key pair that can be used for online signature generation. This scheme is computationally heavy due to the usage of RSA-based digital signature and also incurs heavy communication costs. In addition, it does not provide anonymity or untraceability and is vulnerable to DoS attacks, offline guessing attacks, ESL attacks, and impersonation attacks. It has no support for blockchain technology.

Eddine *et al.* [130] proposed the EASBF scheme for the Internet-of-Vehicles application

with the On-board Unit (OBU) authenticating with the Blockchain Manager (BM) in order to communicate with the other authenticated OBUs and the fog servers in the same fog area. An Authentication Manager (AM) maintains the results of these authentication procedures. The scheme is designed with elliptic curve cryptography (ECC) and one-way hash functions. This scheme has an average computational performance and low communication cost but lacks anonymity, untraceability, and dynamic node addition and is exposed to stolen smart card, stolen mobile device, DoS, privileged insider, and ESL attacks.

Fan *et al.* [133] developed an improved authentication scheme for generalized IoT applications consisting of smart IoT nodes, base stations, application servers, and blockchain-based on bilinear pairings. This scheme has a very heavy computation cost and comparable communication cost but does not support anonymity, untraceability, and dynamic node addition and is exposed to smart card attacks, stolen mobile device attacks, ESL attacks, DoS attacks, and offline guessing attacks.

Panda *et al.* [253] proposed an authentication scheme using one-way hash chains for multi-level architecture for IoT applications consisting of the device layer, fog layer, and cloud layer. It allows k^{th} device to communicate with l^{th} device using N key pairs per device and a license issued by the Access Managing Node (AMN). This computation and communication costs depend on the number of key pairs per device N. It does not support anonymity, untraceability, and the addition of nodes dynamically and is vulnerable to ESL attacks, offline guessing attacks, stolen smart card/ stolen mobile device attacks, impersonation attacks, and privileged insider attacks.

3.9 Industry trends in smart farming

This section discusses the state-of-the-art development of smart farming from the industry perspective in terms of the applications currently existing and their usage. KhethiNext [15] is an IoT-based service-oriented platform by Pals Agri eCONNECT Private Limited from Hyderabad, India, that provides the farmers access to real-time farm data, helps them connect to the manufacturers, suppliers, and financial institutions and potential buyers while achieving traceability of all the activities. It has been successful in providing higher remuneration to eight farmer producer organizations in three states. IoT4Ag Center [36] at the University of Pennsylvania has planned 28 projects to create the future of farming in the three thrusts agricultural sensor systems, communication and energy systems, and agricultural response systems. Infosys has collaborated with Industrial Internet Consortium (IIC)

to develop Infosys Precision Crop Management Testbed [31] that can create a farm footprint by analyzing IoT data about the farm elevation and contour mapping, soil mapping, crop yield mapping, and assessment of farm productivity partnered with Sakata Seeds Inc. SM4RT TANI platform [35] developed using SigFox's Zero-G network [19] is used by many Malaysian IoT providers and connectivity solutions to allow the farmers to remotely monitor real-time data related to pH, weather, pest infestation and crop condition with the help of sensors that are viewed using SATU dashboard and connected to a mainframe system. This system is combined with a digital management platform for farming called Urus Tani. To the best of our knowledge, efforts to develop security protocols in such smart systems for agriculture have not been ventured into by the industry.

National e-Governance Plan Agriculture (NeGP-A) [7] is an initiative taken up by the Govenment of India (GOI) to apply Information and Communication Technologies in the governance of agriculture through timely information access to crop-cycle, certification and licensisng of products, and farming advisory services with farmer-centric service. Crop-related information specific to the location, season is updated periodically and provided to the farmer. It collaborates with private sector companies for marketing and post-harvest activities. In addition, DACNET project is initiated to facilitate "Agriculture-on-line" with high speed, low error delivery. SeedNet India is online portal to provide all details regarding the ssed industry. Kisan Call Centres have been established to provide support to the farmers via telephone calls. GOI has developed various mobile and web applications such as Kisan Suvidha, Pusha Krishi, Agri Market, mKisan portal, Participatory Guarantee System of India to support farmers in advisory on crops, latest farming technologies, marketing of crops, and direct connection with scientists and experts. Unified Farmer Service Platform (UFSP) interoperates public and private firms to develop IT ecosystems for agriculture. A nationwide database for farmers is underway to allow the farmer access to various schemes and their benefits. The proposed authentication schemes can be integrated such existing IT farming ecosystems to shield them from cyberattacks. The Government of India has also undertaken an initiative to educate and train craftsmen with a competency based curriculum to install, maintain, and troubleshoot the devices and network as an IoT Technician in Smart Agriculture [252].

Blockchain technology is being widely adopted in various fields by the Government of India with the establishment of "Centre of Excellence in Blockchain Technology" [251] in order to create an ecosystem of blockchain solutions and facilitate a coordinated sharing of resources. Coffee Board of India relies on Ethereum based smart contracts for supply chain

tracing of their products to the farmers and favors the farmers with better prices [171]. Tea Board of India has also announced to follow blockchain technology for traceability of their products in the supply chain [268]. The State Government of Maharashtra also plans to adopt blockchain technology in agriculture marketing, vehicle registration, document management systems, and supply chain [121].

3.10 Summary

The review undertaken in this thesis aims to fathom the depth and breadth of the need and application of cryptographic security in the area of Agriculture 4.0. The survey identifies the applications of IoT in agriculture, its benefits, and the attacks and possible remedies. A number of existing testbeds for smart agriculture are studied. Various security protocols have been studied in the subsectors of cybersecurity applicable in agriculture.

It can be understood that research in the design of authentication protocols in the area of smart farming remains stunted even though a wide range of testbeds have been studied, developed, and implemented. The survey on the authentication protocols based on blockchain in smart farming and other IoT-based areas finds that very few blockchain-based solutions have been developed for smart farming and they are insufficent in achieving the required security. This leads to the conclusion that there is an immediate necessity to focus on developing authentication protocols before any message exchange takes place in a smart farming environment. Similarly, the areas of cyber-physical searching, secure encrypted searching, and access control are largely unexplored, and research remains in the infant stages as of now. Supply chain traceability and data aggregation are relatively more researched areas in smart farming.

Chapter 4

Signature-Based Anonymous User Authentication in an IoT-enabled Intelligent Precision Agricultural Environment

In an IoT-enabled intelligent precision agricultural environment, various entities (IoT-enabled smart (sensor) devices, users, and controller nodes acting as gateway nodes) typically communicate over public (insecure) channels. Public channels provide an opportunity for an adversary (either passive or active) to eavesdrop on the communicated messages and delete, modify or inject malicious message contents during their communications. Furthermore, the adversary can also launch several potential attacks, such as "replay, impersonation, man-in-the-middle, Denial-of-Service (DoS), ephemeral secret leakage (ESL), and privileged-insider attacks". Among various security services, user authentication is a promising security solution. It allows an external user with his/her mobile device to securely access the real-time information directly from the deployed smart devices in the IoT-enabled intelligent precision agricultural environment. User authentication is essential because the collected data at the controller nodes from their attached smart devices need not always be the live and real-time data. In this chapter, we attempt to design a novel signature-based three-factor user authentication scheme in such an environment, with a user's password, biometric information, and mobile devices being the three factors in the scheme.

4.1 System models

In this section, we discuss the related network and threat models utilized in the proposed "signature-based three-factor user authentication scheme".

4.1.1 Network model

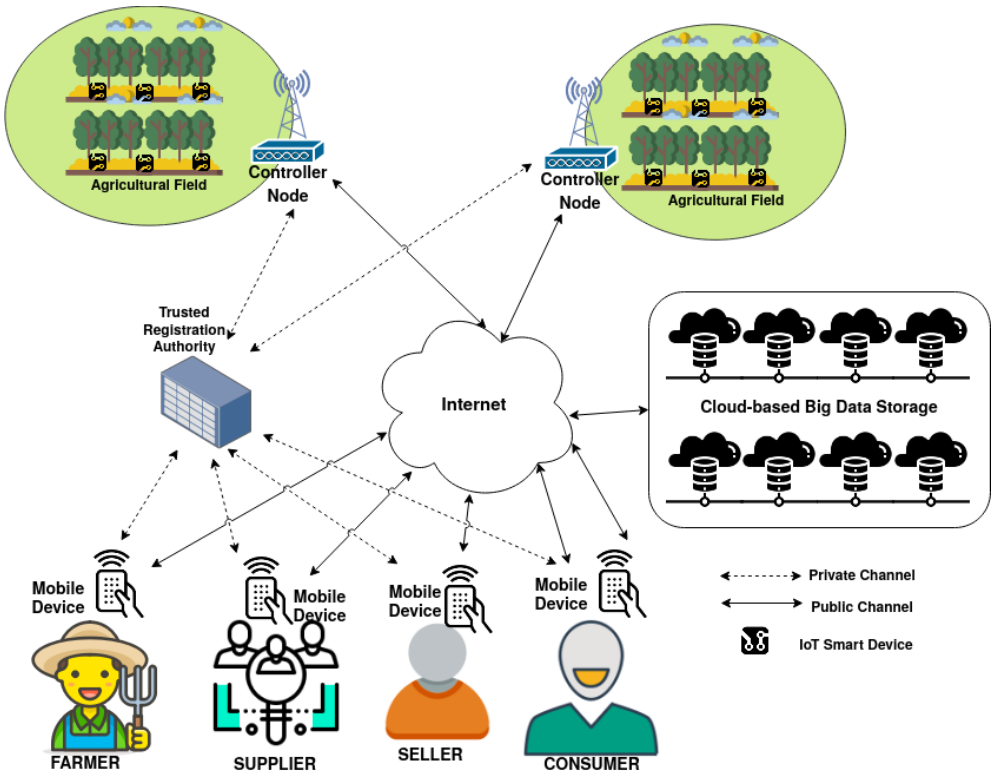

Figure 4.1: Cloud-based IoT-enabled intelligent precision agricultural environment with big data storage

The architecture for a cloud-based IoT-enabled intelligent precision agricultural environment is represented in Figure 4.1. There are multiple agricultural fields, each having a number of IoT smart devices and a controller node. These are monitored by a number of interested users, such as farmers, suppliers, sellers, and customers, with the help of mobile devices. A fully trusted registration authority (TRA) registers the IoT smart devices, the controller nodes, and the users via a secure private channel during their enrollment phases. A registered user must first mutually authenticate with the controller node of the field and agree on a shared secret key. The controller node then contacts the associated accessed IoT

smart device to perform a mutual authentication that ends in the establishment of a common session key between the user and the IoT smart device. Thereafter, the IoT smart devices placed in various positions in the field observe the environmental parameters and send the data to the interested users using the established session keys. In addition, the controller nodes can also securely access the sensing data from the IoT smart devices through the authentication process and forward the received sensor data to the cloud encrypted using the public key of that cloud server. The sensor data is then decrypted by the public key of the cloud server in order to be stored on a cloud-based big data storage system. The authentic and genuine data received from the controller node can be utilized for Big Data analytics in order to analyze the best possible measure to be taken to improve or sustain the state of the field. The Big Data applications in the smart farming scenario are not rigorously about key production. However, these applications have a crucial role in order to improve the efficiency of the whole supply chain and alleviate food security concerns too.

4.1.2 Threat model

The threat models "Dolev-Yao (DY) threat model" [125] along with the contemporary *de facto* "Canetti and Krawczyk's model (CK-adversary model)" [94] discussed in Section 1.4.1 of Chapter 1 are applied. The end-point entities (IoT smart devices and users) are untrustworthy entities in the network. An adversary A is assumed to have the following capabilities:

1) $\mathcal{A}$ can seize, remove, modify and re-transmit existing messages or circulate counterfeit messages for any communication in public channels between IoT smart devices, user mobile devices, and controller node.

2) Impersonation of IoT smart devices or user mobile devices or the controller nodes may be carried out by $\mathcal{A}$ to perform actions on their behalf.

3) Simultaneous multiple executions of the protocol may be initialized by $\mathcal{A}$. The IoT smart devices, user mobile devices, and controller nodes may take part in any number of such concurrent executions at the same time.

4) The smart IoT devices, user mobile devices, and controller nodes are honest and stateless. $\mathcal{A}$ is stateful.

5) Hijacking of session states during communication among smart devices, user mobile devices, and controller nodes may be employed by $\mathcal{A}$ to extract secret credentials.

In addition, physical capture of smart IoT devices and user mobile devices is employed by $\mathcal{A}$ using power analysis attacks [200, 235] and timing attacks [199] to extract secret credentials from their memory. The adversary cannot compromise the controller nodes since they are put under a physical locking system as suggested in [75, 335]. The secret credentials in the controller nodes will be stored in their secure databases to prevent the stolen verifier attack. Thus, the adversary will have no access to the credentials stored in the controller nodes through the stolen verifier attack, so no other attacks, such as controller node impersonation attacks, can be launched through the stolen verifier attacks.

4.2 Research contributions

The following are the primary research contributions from this work:

- A new signature-based three-factor user authentication scheme has been proposed in an IoT-enabled intelligent precision agricultural environment. The proposed scheme is based on "elliptic curve cryptography (ECC)" techniques and signatures. Apart from these, the proposed scheme uses three factors (user password, biometrics, and mobile device) and the widely-accepted fuzzy extractor method [124] for user biometric purposes in order to avoid Denial-of-Service (DoS) attacks as compared to other biometric verification techniques, such as biohashing [69, 184, 223]. The proposed scheme also provides several functionality features, such as the "user mobile device revocation phase" and the "dynamic IoT smart device addition phase".

- The proposed scheme is robust against a variety of potential attacks that are needed in an IoT-enabled intelligent precision agricultural environment, which is shown through the "formal security analysis using the broadly-used Real-Or-Random (ROR) model" [94], the formal security verification using the popular adopted software validation tool known as "Automated Validation of Internet Security Protocols and Applications (AVISPA)" [9] and informal (non-mathematical) security analysis.

- The testbed experimentation of various cryptographic primitives is performed using the widely-accepted "Multiprecision Integer and Rational Arithmetic Cryptographic Library (MIRACL)" [38] to measure the average time needed for executing the primitives. These experimental results are used in computing the computational time needed for the proposed scheme and other existing competing user authentication schemes.

- A detailed comparative study among the proposed scheme and other existing competing user authentication schemes shows that the proposed scheme has "a better trade-off among its offered security and functionality features, and communication and computational overheads as compared to those for other competing schemes".

4.3 The proposed user authentication protocol

In this section, we provide the design of a new "three-factor user authentication scheme" in an IoT-enabled intelligent precision agricultural environment, which relies on a fuzzy extractor technique for a user's biometric verification, elliptic curve cryptography (ECC) technique, and signature. The proposed scheme is based on the network model shown in Figure 4.1. The proposed scheme has the following phases:

- **System initialization phase**: This phase is executed by the TRA in order to select all the system parameters for the involved entities (users, controller nodes, and IoT smart devices), including the one-way cryptographic hash function $H(\cdot)$, fuzzy extractor functions and elliptic curve parameters.

- **Enrollment phase**: In this phase, the TRA registers all the deployed IoT smart devices in their respective agriculture field (also called the deployment area), controller nodes in their corresponding deployment area, and the users who are provided authorized access to the real-time sensing data directly from the designated IoT smart devices in the deployment area(s).

- **Login and authenticated key agreement phase**: This phase allows a registered user to authenticate with both the controller node and the designated accessed IoT smart devices. After mutual authentication, the user establishes session keys with the designated IoT smart devices in order to make secure communication of the real-time sensing information.

- **User mobile device revocation phase**: If a user's registered mobile device is lost or stolen by an adversary, this phase allows the same user to re-issue a new mobile device by requesting the TRA via the secure channel.

- **Dynamic IoT smart device addition phase**: This phase becomes necessary when some IoT smart devices are physically captured by an adversary (according to the threat model provided in Section 4.1.2) or because of exhaustion of battery power.

Thus, this phase permits new IoT smart devices to be deployed in the existing network even after the initial deployment of nodes.

- **User biometric and password change phase**: This phase is essential to keep the system's security at the highest level. It enables a user to change his/her credentials, such as his/her password and even long-standing biometrics, at any time. This phase is executed locally without any more contact with the TRA to minimize the communication and computational overheads.

To protect against "replay attacks", we utilize both the current timestamps and random nonces. In order to serve this goal, all the entities in the network are assumed to be synchronized with their clocks. This is a typical assumption that is applied in designing various security protocols in networks [100, 114, 115, 116, 127, 331, 332, 334]. Various notations and their meanings tabulated in Table 4.1 are applied to describe and analyze the proposed scheme's phases.

4.3.1 System initialisation phase

The following steps are involved in this phase:

- *Step* S_1: The TRA selects a "non-singular elliptic curve $E_q(a, b)$ of the type: $y^2 = x^3 + ax + b \pmod q$ over the Galois field $GF(q)$, with a point at infinity (zero point) $\mathcal{O}$, constants $a, b \in Z_q = \{0, 1, 2, \cdots, q-1\}$ such that $4a^3 + 27b^2 \neq 0 \pmod q$" is satisfied, q is a sufficiently large prime so that the computational "elliptic curve discrete logarithm problem (ECDLP)" and "elliptic curve decisional Diffie-Hellman problem (ECDDHP)" become intractable. The TRA then picks a base point $G \in E_q(a, b)$ whose order n_G is as large as q, that is, $n_G \cdot G = G + G + \cdots + G$ (n_G times) $= \mathcal{O}$, the point at infinity or zero point, where $n_G \cdot G$ represents the elliptic curve point (scalar) multiplication.

- *Step* S_2: The TRA picks a "collision-resistant one-way cryptographic hash function", say $H(\cdot)$ (for instance, SHA-256 hash algorithm may be used [232]).

- *Step* S_3: For user biometric verification, we use the widely-accepted fuzzy extractor [124]. A fuzzy extractor has the following two important functions:

  - *Gen*: It is a "probabilistic generation function" that takes user's biometrics Bio_U as input string and produces an l_b-bit extracted string, say $\sigma_i \in \{0, 1\}^{l_b}$ as the

Table 4.1: Notations and their description

Notation	Significance
TRA	Trusted Registration Authority
$E_q(a,b)$	A non-singular elliptic curve of the form: $y^2 = x^3 + ax + b \pmod q$ with $4a^3 + 27b^2 \neq 0 \pmod q$
G	A base point in $E_q(a,b)$ of order n_G as big as q
$x \cdot G$	An elliptic curve point multiplication: $x \cdot G = G + G + \cdots + G$ (x times)
$P + Q$	Elliptic curve point addition; $P, Q \in E_q(a,b)$
RTS_X	Registration timestamp issued by the TRA to an entity X
CN	Controller node
UD, MD_U	User and its mobile device, respectively
SN	IoT smart (sensor) device
ID_C, TID_C, RID_C	CN's real identity, temporary identity, and pseudo-identity, respectively
ID_U, TID_U, RID_U, Pwd_U, Bio_U	UD's real identity, temporary identity, pseudo-identity, password and biometric template, respectively
ID_S, TID_S, RID_S, TC_S	SN's real identity, temporary identity, and pseudo-identity and temporal credential, respectively
pr_{TRA}, Pub_{TRA}	Private and public key of TRA, respectively
pr_C, Pub_C	Private and public keys of CN, respectively
pr_U, Pub_U	Private and public keys of UD, respectively
pr_S, Pub_S	Private and public keys of SN, respectively
r_1, r_2, r_3, y_1	UD's random secrets
y_2, y_4	CN's random secrets
y_3, r_4	SN's random secrets
$Enc_K(\cdot)$, $Dec_K(\cdot)$	Symmetric encryption and decryption functions using the shared key K
$\|\|$	Concatenation operation
T_X	Current timestamp generated by an entity X
$*$	Modular multiplication in a finite field Z_q
$\oplus$	Exclusive-OR (XOR)s operation
ΔT	Maximum transmission delay related to a message
$h(\cdot)$	"Collision-resistant cryptographic one-way hash function"
$Gen(\cdot), Rep(\cdot)$	Fuzzy extraction generation and reproduction functions, respectively
σ_U, τ_U	Biometric secret and public parameters, respectively
et	"Error tolerance threshold of fuzzy extractor $Rep(\cdot)$ function"

"user biometric secret key" and τ_i as an "auxiliary string (also called a reproduction public parameter)", that is, $Gen(Bio_U) = (\sigma_U, \tau_U)$.

– *Rep*: This is a "deterministic function" that recovers the "original biometric secret key σ_U" with the help of the helper data, i.e., public reproduction parameter τ_U and biometrics input, say Bio'_U, that is, $Rep(Bio'_U, \tau_U) = \sigma_U$ provided that the "Hamming distance between Bio_U and Bio'_U is less or equal to a pre-defined error-tolerance threshold value et".

- *Step* S_4: The TRA also picks symmetric encryption and decryption algorithms, say $Enc_K(\cdot)$ and $Dec_K(\cdot)$, that use the shared key K. For example, we may use Advanced Encryption Standard (AES-128) symmetric cipher [8] that uses 128-bit plaintext block and outputs the corresponding 128-bit ciphertext block.

- *Step* S_5: Finally, the TRA picks its own private key pr_{TRA} in Z_q^* and computes the respective public key $Pub_{TRA} = pr_{TRA}.G$, and publishes the public key Pub_{TRA} and all other domain parameters $\{E_q(a, b), G, H(\cdot), Gen(\cdot), Rep(\cdot), et, Enc_K(\cdot), Dec_K(\cdot)\}$ as public.

4.3.2 Enrollment phase

In this phase, we discuss the enrollment of each IoT smart device, the controller nodes, and the user registration. The detailed discussion is provided in the following subsections.

1) **IoT smart device enrollment:** This phase is executed in offline mode by the TRA to register/enroll all the IoT smart device (SN) in a particular zone in the agriculture field containing their controller node (CN). The following are the steps behind this process:

- *Step* SDE_1: TRA picks a real identity ID_S, temporary identity TID_S, and a random secret $s_1 \in Z_q^*$ for each SN. TRA also computes a pseudo-identity for each SN as $RID_S = H(ID_S || s_1)$.

- *Step* SDE_2: TRA picks a random private key $pr_S \in Z_q^*$ and computes the corresponding public key $Pub_S = pr_S \cdot G$. and a temporal credential for each SN as $TC_S = H(RID_S || pr_S || pr_{TRA} || RTS_S)$ where RTS_S is the current timestamp of registration of SN.

- Step SDE_3: TRA finally preloads SN with the credentials $\{(RID_S, TID_S, TC_S), H(\cdot), E_q(a,b), G, (pr_S, Pub_S)\}$ in its memory. In addition, TRA publishes Pub_S as SN's public key.

The IoT smart device registration phase is then briefed in Figure 4.2.

Trusted Registration Authority (TRA)	IoT Smart Device (SN)
Pick ID_S, TID_S, $s_1 \in Z_q^*$.	
Compute $RID_S = H(ID_S \| s_1)$.	
Pick $pr_S \in Z_q^*$.	
Calculate $Pub_S = pr_S \cdot G$,	
$TC_S = H(RID_S \| pr_S \| pr_{TRA} \| RTS_S)$.	
Preload SN with	$\{(RID_S, TID_S, TC_S),$
$\{(RID_S, TID_S, TC_S), H(\cdot),$	$H(\cdot), E_q(a,b), G, (pr_S, Pub_S)\}$
$E_q(a,b), G, (pr_S, Pub_S)\}$	are in its memory.

Figure 4.2: Summary of IoT smart device registration phase

2) **User registration:** This phase is also executed by the TRA" through a "secure channel (for example, in-person) because user registration is a one-time process". The following are the steps needed in this phase:

- *Step* UR_1: The user UD picks random secrets r_1, $r_2 \in Z_q^*$, user real identity ID_U and user password Pwd_U. UD then imprints the biometric template Bio_U at the sensor of his/her own mobile device MD_U.
- *Step* UR_2: UD computes pseudo-identity $RID_U = H(ID_U \| r_1)$ and pseudo-password $RPW_U = H(Pwd_U \| r_1)$. UD then sends the registration request $Msg_{UR_1} = \langle RID_U, RPW_U \oplus r_2 \rangle$ to the TRA via secure channel.
- *Step* UR_3: TRA checks if RID_U exists in its database after receiving the request Msg_{UR_1}. If it does not exist, TRA selects temporary user identity TID_U and computes a parameter $\alpha_U = H(RID_U \| pr_{TRA} \| RTS_U) \oplus (RPW_U \oplus r_2)$ where RTS_U is the current timestamp of user registration. TRA then sends $Msg_{UR_2} = \langle \alpha_U, TID_U \rangle$ to the user UD via a secure channel.
- *Step* UR_4: The user UD computes $\beta_U = \alpha_U \oplus r_2$ which yields $H(RID_U \| pr_{TRA} \| RTS_U) \oplus RPW_U$ from α_U of the received message Msg_{UR_2}. UD applies the "fuzzy

extractor generation function $Gen(\cdot)$ on the biometric data Bio_U" to produce the biometric secret key σ_U and public reproduction parameter τ_U, that is, $Gen(Bio_U) = (\sigma_U, \tau_U)$.

- *Step* UR_5: UD also selects his/her own private key as $pr_U \in Z_q^*$ and calculates the corresponding public parameter as $Pub_U = pr_U \cdot G$. The private key pr_U is hidden as $pr_U^* = pr_U \oplus H(ID_U || Pwd_U || \sigma_U)$, β_U is hidden as $\beta_U^* = \beta_U \oplus H(r_1 || \sigma_U || Pwd_U)$, r_1 is also hidden as $r_1^* = r_1 \oplus H(Pwd_U || \sigma_U || ID_U)$.
- *Step* UR_6: UD also computes a verifier $Ver_U = H(r_1 || Pwd_U || pr_U || \beta_U || \sigma_U || ID_U || \tau_U)$ and encrypts $\{(TID_U, RID_U), pr_U^*, \beta_U^*, r_1^*\}$ with $H(Pwd_U || ID_U || \sigma_U)$ as the key. The user mobile device MD_U stores the credentials $\{Enc_{H(Pwd_U||ID_U||\sigma_U)}[(TID_U, RID_U), pr_U^*, \beta_U^*, r_1^*], Ver_U, \tau_U, H(\cdot), E_q(a,b), G, Gen(\cdot), Rep(\cdot), et\}$, where et is the pre-defined error tolerance threshold value used in the $Rep(\cdot)$ fuzzy extractor function. In addition, UD publishes Pub_U as the public key.

The user registration phase is briefed in Figure 4.3.

3) **Controller node enrollment:** In this phase, each controller node CN is enrolled by the TRA prior to their placement in the agricultural zones. We assume that a set of users UD along with a set of IoT smart devices SN are associated with a particular CN. The enrollment of a specific CN has the following steps:

- *Step* CNE_1: TRA picks a real identity for the controller node as ID_C, its temporary identity as TID_C, and a random secret $c_1 \in Z_q^*$. TRA computes its pseudo-identity as $RID_C = H(ID_C || c_1)$.
- *Step* CNE_2: TRA preloads the controller node CN with the credentials $\{(RID_C, TID_C), \{(RID_U, TID_U)\}, \{(RID_S, TID_S, TC_S)\}, H(\cdot), E_q(a,b), G\}$.
- *Step* CNE_3: Later, CN picks its own random private key $pr_C \in Z_q^*$ and computes the corresponding public key as $Pub_C = pr_C \cdot G$.
- *Step* CNE_4: CN adds its private and public key pair (pr_C,Pub_C) to its tamper-proof secure memory database. Thus, the credentials $\{(RID_C, TID_C), \{(RID_U, TID_U)\}, \{(RID_S, TID_S, TC_S)\}, (pr_C, Pub_C), H(\cdot), E_q(a,b), G\}$ are stored in the tamper-proof secure memory database so that the stolen verifier attack can be prevented by an attacker in order to launch other attacks, including the controller node impersonation attack. In addition, CN also publishes Pub_C as its public key.

User(UD)		Trusted Registration Authority (TRA)
Pick random secret numbers r_1, $r_2 \in Z_q^*$. Enter real identity ID_U and choose password Pwd_U. Imprint personal biometrics Bio_U at MD_U. Compute $RID_U = H(ID_U \| r_1)$, $RPW_U = H(Pwd_U \| r_1)$.		
	$\xrightarrow{Msg_{UR_1} : \langle RID_U, RPW_U \oplus r_2 \rangle}$ (via secure channel)	
		Check if RID_U exists in its database. If not, pick a new temporary identity TID_U corresponding to RID_U. Compute $\alpha_U = H(RID_U \| pr_{TRA} \| RTS_U) \oplus (RPW_U \oplus r_2)$.
	$\xleftarrow{Msg_{UR_2} : \langle \alpha_U, TID_U \rangle}$ (via secure channel)	
Calculate $\beta_U = \alpha_U \oplus r_2$ $= H(RID_U \| pr_{TRA} \| RTS_U) \oplus RPW_U$, $Gen(Bio_U) = (\sigma_U, \tau_U)$. Pick private key $pr_U \in Z_q^*$ and compute $Pub_U = pr_U \cdot G$. Calculate $pr_U^* = pr_U \oplus H(ID_U \| Pwd_U \| \sigma_U)$, $\beta_U^* = \beta_U \oplus H(r_1 \| \sigma_U \| Pwd_U)$, $r_1^* = r_1 \oplus H(Pwd_U \| \sigma_U \| ID_U)$, $Ver_U = H(r_1 \| Pwd_U \| pr_U \| \beta_U \| \sigma_U \| ID_U \| \tau_U)$. $Enc_{H(Pwd_U \| ID_U \| \sigma_U)}[(TID_U, RID_U), pr_U^*, \beta_U^*, r_1^*]$. Store $\{Enc_{H(Pwd_U \| ID_U \| \sigma_U)}[(TID_U, RID_U)$, $pr_U^*, \beta_U^*, r_1^*]$, Ver_U, τ_U, $H(\cdot)$, $E_q(a,b)$, G, $Gen(\cdot)$, $Rep(\cdot)$, $et\}$ in MD_U.		

Figure 4.3: Summary of user registration phase

The overall CN enrollment process is briefed in Figure 4.4.

4.3.3 Login and authenticated key agreement phase

This phase allows a registered legal user UD with his/her mobile device MD_U having legitimate credentials to log into the system by entering his/her credentials and verifying that the user UD is legitimate. This process is executed between three entities, which are the user UD, a controller node CN under which UD is already registered, and a designated accessed IoT smart device SN from which UD wants to access the "real-time sensing information directly". The following are the essential steps:

- *Step* LA_1: UD enters identity ID_U, password Pwd_U, and biometric template Bio'_U

Trusted Registration Authority (TRA)	Controller Node(CN)
Pick ID_C, TID_C, $c_1 \in Z_q^*$. Compute $RID_C = H(ID_C \|\| c_1)$. Preload the controller node CN with $\{(RID_C, TID_C), (RID_U, TID_U)$, $(RID_S, TID_S, TC_S), H(\cdot), E_q(a,b), G\}$.	
	Pick private key $pr_C \in Z_q^*$. Compute public key $Pub_C = pr_C \cdot G$. Store the credentials $\{(RID_C, TID_C)$, $(RID_U, TID_U), (RID_S, TID_S, TC_S)$, $(pr_C, Pub_C), H(\cdot), E_q(a,b), G\}$ in secure memory (database).

Figure 4.4: Summary of controller node registration phase

at the sensor of the mobile device MD_U. MD_U retrieves the biometric secret key σ_U by applying the fuzzy extractor reproduction function as $Rep(Bio'_U, \tau_U) = \sigma_U$ provided that the "Hamming distance between the registered biometric template Bio_U and current biometric template Bio'_U is less than or equal to et". MD_U then calculates $H(Pwd_U||ID_U||\sigma_U)$ to decrypt and retrieve $\{(TID_U, RID_U), pr_U^*, \beta_U^*, r_1^*\} = Dec_{H(Pwd_U||ID_U||\sigma_U)}\ [Enc_{H(Pwd_U||ID_U||\sigma_U)}\ [(TID_U, RID_U), pr_U^*, \beta_U^*, r_1^*]]$.

- *Step* LA_2: MD_U calculates $r_1 = r_1^* \oplus H(Pwd_U|| \sigma_U||ID_U)$, $\beta_U = \beta_U^* \oplus H(r_1|| \sigma_U|| Pwd_U)$ and $pr_U = pr_U^* \oplus H(ID_U|| Pwd_U|| \sigma_U)$. It also re-computes the verifier $Ver'_U = H(r_1 ||Pwd_U ||pr_U ||\beta_U ||\sigma_U ||ID_U ||\tau_U)$ and checks $Ver'_U \stackrel{?}{=} Ver_U$. If the check is true, the user UD is verified as a genuine user at MD_U and his/her identity ID_U, password Pwd_U, and biometric template Bio'_U are legitimate. Otherwise, this phase is immediately terminated by the user UD.

- *Step* LA_3: UD now enters the pseudo-identity of the accessed smart sensor node SN as RID_S from his/her mobile device application program interface (API). MD_U then generates random secrets r_3, $y_1 \in Z_q^*$ and the current timestamp T_1 to compute $RID_U^* = H(RID_U||T_1)$, $M_1 = RID_U^* \oplus (r_3|| RID_S)$, and $Y_1 = H(y_1|| \sigma_U|| RID_U|| Pwd_U) \cdot G$. It also generates the signature on r_3 and y_1 using the private key pr_U as $Sign_{y_1} = H(y_1|| \sigma_U|| RID_U|| Pwd_U) + H(r_3|| M_1|| RID_U|| RID_S|| T_1||Pub_U) * pr_U \pmod q$.

UD then sends the message $Msg_1 = \langle\ TID_U,\ M_1,\ Y_1,\ Sign_{y_1}, T_1\rangle$ to the controller node CN via an open channel.

- *Step* LA_4: After receiving Msg_1 from the user UD at time T_1^*, the controller node CN first verifies the timestamp as $|T_1^* - T_1| < \Delta T$, where ΔT is the "maximum transmission delay associated with the message". If it is valid, CN checks the existence of TID_U in its database and retrieves the corresponding RID_U to calculate $RID_U^* = H(RID_U||T_1)$, and decrypts M_1 to obtain $(r_3||\ RID_S) = M_1 \oplus RID_U^*$. CN verifies the signature $Sign_{y_1}$ to confirm the sender and verify if the parameters have been received correctly by the condition: $Sign_{y_1} \cdot G \stackrel{?}{=} Y_1 + H(r_3||\ M_1||\ RID_U||\ RID_S||\ T_1||Pub_U) \cdot Pub_U$. If the signature is valid, UD is considered as an authentic user by CN.

- *Step* LA_5: CN generates a random secret $y_2 \in Z_q^*$ and the current timestamp T_2. It computes a common secret $M_2 = H(r_3||\ RID_S||\ RID_U||\ TC_S||\ T_2) \oplus H(RID_S||\ TC_S||\ T_2)$, $Y_2 = H(y_2||\ RID_C||\ pr_C) \cdot G$ and the signature on y_2 as $Sign_{y_2} = H(y_2||\ RID_C||\ pr_C) + H(M_2||\ RID_S||\ TID_C||\ T_2\ ||Pub_C||Y_1) * pr_C \pmod q$. CN sends the message $Msg_2 = \langle\ TID_C,\ M_2,\ Y_1,\ Y_2,\ Sign_{y_2},\ T_2\rangle$ to the smart device node SN via public channel.

- *Step* LA_6: Once the smart device node SN receives the message Msg_2 at time T_2^*, it verifies the timestamp as $|T_2^* - T_2| < \Delta T$. If so, it verifies the controller node signature by checking the condition: $Sign_{y_2} \cdot G \stackrel{?}{=} Y_2 + H(M_2||\ RID_S||\ TID_C||\ T_2\ ||Pub_C||Y_1) \cdot Pub_C$. If the signature is verified correctly, CN is considered as authentic entity by SN, and retrieves the common secret as $H(r_3||\ RID_S||\ RID_U||\ TC_S||\ T_2) = M_2 \oplus H(RID_S||\ TC_S||\ T_2)$.

- *Step* LA_7: SN generates two random secrets y_3, $r_4 \in Z_q^*$ and the current timestamp T_3. It generates and encrypts the second common secret as $M_3 = H(r_4||\ RID_S||\ TC_S||\ T_3) \oplus H(H(y_3||\ RID_S||\ pr_S) \cdot Y_1||\ RID_S||\ T_3)$, and computes $Y_3 = H(y_3||\ RID_S||\ pr_S) \cdot G$. It then computes the common session key shared with the user UD as $SK_{SU} = H(H(r_3||\ RID_S||\ RID_U||\ TC_S||\ T_2)\ ||H(r_4||\ RID_S||\ TC_S||\ T_3)\ ||H(y_3||\ RID_S||\ pr_S) \cdot Y_1)$ along with its verifier $M_4 = H(SK_{SU}||\ T_3)$. It also generates the signature on y_3 and r_4 as $Sign_{y_3} = H(y_3||\ RID_S||\ pr_S) + H(M_3||M_4||\ RID_S||\ TID_C||\ T_3||Pub_S) * pr_S \pmod q$. It generates a new temporary identity TID_S^{new}, encrypts it as $TID_S^* = TID_S^{new} \oplus H(H(y_3||\ RID_S||\ pr_S) \cdot Y_2||\ TID_S||TC_S||\ T_3)$ and updates TID_S with TID_S^{new} in its database. The smart device node SN sends the message $Msg_3 = \langle TID_S^*,\ M_3,\ M_4,\ Y_3,$

$Sign_{y_3}$, T_3 $\rangle$ to the controller node CN via open channel.

- *Step* LA_8: After receiving the message Msg_3 from the smart device SN at time T_3^*, CN verifies the timestamp as $|T_3^* - T_3| < \Delta T$. If the timestamp is valid, CN checks the smart device signature: $Sign_{y_3} \cdot G \stackrel{?}{=} Y_3 + H(M_3||M_4||\ RID_S||\ TID_C||\ T_3||\ Pub_S) \cdot Pub_S$. If the signature validation is successful, it also retrieves the new temporary identity of SN as $TID_S^{new} = TID_S^* \oplus H(H(y_2||\ RID_C||\ pr_C) \cdot Y_3||\ TID_S||TC_S||\ T_3)$ and updates TID_S with TID_S^{new} in its secure database.

- *Step* LA_9: CN generates another random secret $y_4 \in Z_q^*$ and the current timestamp T_4. CN computes $Y_4 = H(y_4||\ RID_C||\ pr_C) \cdot G$ and $M_5 = H(r_3||\ RID_S||\ RID_U||\ TC_S||\ T_2) \oplus H(H(y_4||RID_C||pr_C) \cdot Y_1||\ RID_U||\ T_4)$, and generates the signature on y_4 as $Sign_{y_4} = H(y_4||\ RID_C||\ pr_C) + H(M_3||\ M_4||M_5||\ Y_1\ ||Y_3||RID_U||\ T_3||\ T_4||Pub_C) * pr_C \pmod q$. It generates a new temporary identity for user UD as TID_U^{new} and encrypts it as $TID_U^* = TID_U^{new} \oplus H(H(y_4||\ RID_C||\ pr_C) \cdot Y_1||\ TID_U||\ T_4)$ and updates TID_U with TID_U^{new}. CN then sends the message $Msg_4 = \langle TID_U^*, M_3,\ M_4,\ M_5,\ Y_3,\ Y_4,\ Sign_{y_4},\ T_3,\ T_4 \rangle$.

- *Step* LA_{10}: UD receives Msg_4 from the controller node CN at time T_4^* and verifies the timestamp by verifying if $|T_4^* - T_4| < \Delta T$ is true. It then verifies the controller signature as $Sign_{y_4} \cdot G \stackrel{?}{=} Y_4 + H(M_3||\ M_4||M_5||\ Y_1\ ||Y_3||RID_U||\ T_3||\ T_4||Pub_C) \cdot Pub_C$. If these are valid, it retrieves the first common secret as $H(r_3||\ RID_S||\ RID_U||\ TC_S||\ T_2) = M_5 \oplus H(H(y_1||\ \sigma_U||\ RID_U||\ Pwd_U) \cdot Y_4||\ RID_U||\ T_4)$, and second common secret as $H(r_4||\ RID_S||\ TC_S||\ T_3) = M_3 \oplus H(H(y_1||\ \sigma_U||\ RID_U||\ Pwd_U) \cdot Y_3||\ RID_S||\ T_3)$, and computes the common session key as $SK_{US} = H(H(r_3||\ RID_S||\ RID_U||\ TC_S||\ T_2)\ ||H(r_4||\ RID_S||\ TC_S||\ T_3)\ ||\ H(y_1||\ \sigma_U||\ RID_U||\ Pwd_U) \cdot Y_3)$ along with its verifier M_4' $= H(SK_{US}||\ T_3)$. If $M_4' = M_4$, the common session key SK_{US} computed at the user side UD is the same as SK_{SU} computed at the smart device node side, and is stored by the user in its mobile device MD_U. Lastly, MD_U retrieves its temporary identity as $TID_U^{new} = TID_U^* \oplus H(H(y_1||\ \sigma_U||\ RID_U||\ Pwd_U) \cdot Y_4||\ TID_U||\ T_4)$ and updates TID_U with TID_U^{new} in its storage.

- *Step* LA_{11}: The controller node CN now generates its new temporary identity TID_C^{new} and updates TID_C with TID_C^{new} in its secure database. The smart device node SN also stores the common session key SK_{SU} in its storage.

Thus, both the user UD and the smart device SN compute the same common session

key SK_{SU} (= SK_{US}), which is then used for any further communication. The login and authentication phase has been summarized in Figure 4.5.

4.3.4 User mobile device revocation phase

A mobile device needs to be revoked when a user no longer is able to monitor the field because his/her mobile device has been stolen or lost. To accommodate this, the user UD needs to register with a new mobile device, say MD'_U. The following are the steps necessary to achieve this task:

- *Step* MDR_1: UD first picks random secrets r'_1, $r'_2 \in Z_q^*$, user real identity ID'_U and user password Pwd'_U, and imprints the biometric template Bio_U at the sensor of his/her own new mobile device MD'_U.

- *Step* MDR_2: UD computes pseudo-identity $RID'_U = H(ID'_U|| \ r'_1)$ and pseudo-password $RPW'_U = H(Pwd'_U|| \ r'_1)$. UD then sends the registration request $Msg'_{UR_1} = \langle \ RID'_U, \ RPW'_U \oplus r'_2 \ \rangle$ to the TRA via secure channel.

- *Step* MDR_3: TRA checks if RID'_U exists in its database after receiving the request Msg'_{UR_1}. If it does not exist, TRA selects temporary user identity TID'_U and computes a parameter $\alpha'_U = H(RID'_U||pr_{TRA}||RTS'_U) \oplus (RPW'_U \oplus r'_2)$ where RTS'_U is the current timestamp of user registration. TRA then sends $Msg'_{UR_2} = \langle \ \alpha'_U, \ TID'_U \ \rangle$ to the user UD via secure channel.

- *Step* MDR_4: The user computes $\beta_U = \alpha'_U \oplus r'_2$ which yields $H(RID'_U|| \ pr_{TRA}|| \ RTS'_U) \oplus RPW'_U$ from α'_U of the received message Msg'_{UR_2}. UD applies the fuzzy extractor generation function $Gen(\cdot)$ on the biometric data Bio_U to produce the biometric secret key σ_U and public reproduction parameter τ_U as $Gen(Bio_U) = (\sigma_U, \ \tau_U)$.

- *Step* MDR_5: UD selects his/her own private key as $pr'_U \in Z_q^*$ and calculates the corresponding public parameter as $Pub'_U = pr'_U \cdot G$. The private key pr'_U is hidden as $pr^*_U = pr'_U \oplus H(ID'_U|| \ Pwd'_U|| \ \sigma_U)$, β_U is hidden as $\beta^*_U = \beta_U \oplus H(r'_1|| \ \sigma_U|| \ Pwd'_U)$, r'_1 is also hidden as $r^*_1 = r'_1 \oplus H(Pwd'_U|| \ \sigma_U||ID'_U)$.

- *Step* MDR_6: UD computes a verifier $Ver_U = H(r'_1 \ ||Pwd'_U \ ||pr_U \ ||\beta_U \ ||\sigma_U \ ||ID'_U \ ||\tau_U)$ and encrypts $\{(TID'_U, \ RID'_U), \ pr^*_U, \ \beta^*_U, \ r^*_1\}$ with $H(Pwd'_U \ ||ID'_U \ ||\sigma_U)$ as the key. MD_U stores the credentials $\{Enc_{H(Pwd'_U||ID'_U||\sigma_U)}[(TID'_U, \ RID'_U), \ pr^*_U, \ \beta^*_U, \ r^*_1], \ Ver_U,$

User (UD)	Controller Node (CN)	IoT Smart Device Node (SN)
Enter ID_U, Pwd_U, Bio'_U. Compute $Rep(Bio'_U,\tau_U) = \sigma_U$, $Dec_{H(Pwd_U\|\|ID_U\|\|\sigma_U)}[Enc_{H(Pwd_U}$ $\|\|ID_U\|\|\sigma_U)[(TID_U, RID_U)$, $pr_U^*,\ \beta_U^*,\ r_1^*]],\ r_1 = r_1^* \oplus H(Pwd_U\|\|$ $\sigma_U\|\|ID_U)$, $\beta_U = \beta_U^* \oplus H(r_1\|\| \sigma_U\|\| Pwd_U)$, $pr_U = pr_U^* \oplus H(ID_U\|\| Pwd_U\|\| \sigma_U)$, $Ver'_U = H(r_1 \|\|Pwd_U \|\|pr_U \|\|\beta_U \|\|\sigma_U$ $\|\|ID_U \|\|\tau_U)$.If $Ver'_U \stackrel{?}{=} Ver_U$, select RID_S for accessed SN. Generate r_3, $y_1 \in Z_q^*$, T_1. Compute $RID_U^* = H(RID_U\|\|T_1)$, $M_1 = RID_U^* \oplus (r_3\|\| RID_S)$, $Y_1 = H(y_1\|\| \sigma_U\|\| RID_U\|\| Pwd_U) \cdot G$, $Sign_{y_1} = H(y_1\|\| \sigma_U\|\| RID_U\|\| Pwd_U) +$ $H(r_3\|\| M_1\|\| RID_U\|\| RID_S\|\| T_1\|\|Pub_U)$ $*pr_U \pmod q$. $\xrightarrow{Msg_1 = \langle TID_U, M_1, Y_1, Sign_{y_1}, T_1\rangle}$ (via public channel)	Check if $\|T_1^* - T_1\| < \Delta T$? If so, check the existence of TID_U. If so, retrieve corresponding RID_U. Compute $RID_U^* = H(RID_U\|\|T_1)$, $(r_3\|\|RID_S) = M_1 \oplus RID_U^*$. Check if $Sign_{y_1} \cdot G \stackrel{?}{=} Y_1 + H(r_3\|\| M_1\|\|$ $RID_U\|\| RID_S\|\| T_1\|\|Pub_U) \cdot Pub_U$ Generate $y_2 \in Z_q^*$, T_2. Compute $M_2 = H(r_3\|\| RID_S\|\| RID_U\|\|$ $TC_S\|\| T_2) \oplus H(RID_S\|\| TC_S\|\| T_2)$, $Y_2 = H(y_2\|\| RID_C\|\| pr_C) \cdot G$, $Sign_{y_2} = H(y_2\|\| RID_C\|\| pr_C) +$ $H(M_2\|\| RID_S\|\| TID_C\|\| T_2 \|\|Pub_C\|\|Y_1)$ $*pr_C \pmod q$. $\xrightarrow{Msg_2 = \langle TID_C, M_2, Y_1, Y_2, Sign_{y_2}, T_2\rangle}$ (via public channel)	Check if $\|T_2^* - T_2\| < \Delta T$? If so, verify: $Sign_{y_2} \cdot G \stackrel{?}{=} Y_2 + H(M_2\|\| RID_S\|\|$ $TID_C\|\| T_2 \|\|Pub_C\|\|Y_1) \cdot Pub_C$. If valid, compute $H(r_3\|\| RID_S\|\| RID_U\|\| TC_S\|\|$ $T_2) = M_2 \oplus H(RID_S\|\| TC_S\|\| T_2)$. Generate y_3, $r_4 \in Z_q^*$, T_3. Compute $M_3 = H(r_4\|\| RID_S\|\| TC_S\|\|$ $T_3) \oplus H(H(y_3\|\| RID_S\|\| pr_S) \cdot Y_1\|\|$ $RID_S\|\| T_3)$, $Y_3 = H(y_3\|\| RID_S\|\| pr_S) \cdot G$, $SK_{SU} = H(H(r_3\|\| RID_S\|\| RID_U\|\|$ $TC_S\|\| T_2) \|H(r_4\|\| RID_S\|\| TC_S\|\| T_3)$ $\|H(y_3\|\| RID_S\|\| pr_S) \cdot Y_1)$, $M_4 = H(SK_{SU}\|\| T_3)$, $Sign_{y_3} = H(y_3\|\| RID_S\|\| pr_S)+$ $H(M_3\|\|M_4\|\| RID_S\|\| TID_C\|\| T_3\|\|Pub_S)$ $*pr_S \pmod q$.
	Check if $\|T_3^* - T_3\| < \Delta T$? If so, verify: $Sign_{y_3} \cdot G \stackrel{?}{=} Y_3 + H(M_3\|\|M_4\|\| RID_S\|\|$ $TID_C\|\| T_3\|\|Pub_S) \cdot Pub_S$. Compute $TID_S^{new} = TID_S^* \oplus H(H(y_2\|\|$ $RID_C\|\| pr_C) \cdot Y_3\|\| TID_S\|\|TC_S\|\| T_3)$. Update TID_S with TID_S^{new} in its secure database.	Generate TID_S^{new}. Compute $TID_S^* = TID_S^{new} \oplus H(H(y_3\|\| RID_S\|\|$ $pr_S) \cdot Y_2\|\| TID_S\|\|TC_S\|\| T_3)$. Update TID_S with TID_S^{new}. $\xleftarrow{Msg_3 = \langle TID_S^*, M_3, M_4, Y_3, Sign_{y_3}, T_3\rangle}$ (via public channel)
Check if $\|T_4^* - T_4\| < \Delta T$? If so, verify: $Sign_{y_4} \cdot G \stackrel{?}{=} Y_4 + H(M_3\|\| M_4\|\|M_5\|\| Y_1$ $\|\|Y_3\|\|RID_U\|\| T_3\|\| T_4\|\|Pub_C) \cdot Pub_C$ Compute $H(r_3\|\| RID_S\|\| RID_U\|\|$ $TC_S\|\| T_2) = M_5 \oplus H(H(y_1\|\| \sigma_U\|\|$ $RID_U\|\| Pwd_U) \cdot Y_4\|\| RID_U\|\| T_4)$, $H(r_4\|\| RID_S\|\| TC_S\|\| T_3) = M_3 \oplus$ $H(H(y_1\|\| \sigma_U\|\| RID_U\|\| Pwd_U) \cdot Y_3\|\|$ $RID_S\|\| T_3)$, $SK_{US} = H(H(r_3\|\| RID_S\|\| RID_U\|\|$ $TC_S\|\| T_2) \|H(r_4\|\| RID_S\|\| TC_S\|\| T_3)$ $\|H(y_1\|\| \sigma_U\|\| RID_U\|\| Pwd_U) \cdot Y_3)$, $M'_4 = H(SK_{US}\|\| T_3)$ If $M'_4 \stackrel{?}{=} M_4$, $SK_{SU}(= SK_{US})$ is stored at MD_U. Compute $TID_U^{new} = TID_U^* \oplus H(H(y_1\|\|$ $\sigma_U\|\| RID_U\|\| Pwd_U) \cdot Y_4\|\| TID_U\|\| T_4)$. Update TID_U with TID_U^{new} in its storage.	Generate $y_4 \in Z_q^*$, T_4. Compute $Y_4 = H(y_4\|\| RID_C\|\| pr_C) \cdot G$, $M_5 = H(r_3\|\| RID_S\|\| RID_U\|\| TC_S\|\| T_2)$ $\oplus H(H(y_4\|\|RID_C\|\|pr_C) \cdot Y_1\|\| RID_U\|\|$ $T_4)$, $Sign_{y_4} = H(y_4\|\| RID_C\|\| pr_C)+$ $H(M_3\|\| M_4\|\| M_5\|\| Y_1 \|\|Y_3\|\|RID_U\|\| T_3\|\|$ $T_4\|\|Pub_C) *pr_C \pmod q$. Generate TID_U^{new}. Compute $TID_U^* = TID_U^{new} \oplus H(H(y_4\|\|$ $RID_C\|\| pr_C) \cdot Y_1\|\| TID_U\|\| T_4)$. Update TID_U with TID_U^{new}. $\xleftarrow{Msg_4 = \langle TID_U^*, M_3, M_4, M_5, Y_3, Y_4, Sign_{y_4}, T_3, T_4\rangle}$ (via public channel)	
	Generate TID_C^{new}. Update TID_C with TID_C^{new}.	Store the common session key SK_{SU}.
User mobile device and IoT smart device share the common session key SK_{SU} $(= SK_{US})$.		

Figure 4.5: Summary of login and authentication phase

τ_U, $H(\cdot)$, $E_q(a,b)$, G, $Gen(\cdot)$, $Rep(\cdot)$, $et\}$. In addition, UD publishes Pub'_U as the public key.

This phase is briefed in Figure 4.6.

User (UD)	Trusted Registration Authority (TRA)
Pick random secrets r'_1, $r'_2 \in Z_q^*$.	
Select user real identity ID'_U and user password Pwd'_U.	
Imprint biometric template Bio_U at sensor	
of new mobile device MD'_U.	
Compute pseudo-identity $RID'_U = H(ID'_U \|\| r'_1)$	
and pseudo-password $RPW'_U = H(Pwd'_U \|\| r'_1)$.	
$Msg'_{UR_1} = \langle RID'_U, RPW'_U \oplus r'_2 \rangle \xleftarrow{}$	
(via secure channel)	
	Check existence of RID'_U.
	If not, pick temporary user identity TID'_U.
	Compute $\alpha'_U = H(RID'_U \|\| pr_{TRA} \|\| RTS'_U)$
	$\oplus (RPW'_U \oplus r'_2)$.
	$Msg'_{UR_2} = \langle \alpha'_U, TID'_U \rangle \xleftarrow{}$
	(via secure channel)
Compute $\beta_U = \alpha'_U \oplus r'_2$, $Gen(Bio_U) = (\sigma_U, \tau_U)$.	
Select private key $pr'_U \in Z_q^*$.	
Calculate public $Pub'_U = pr'_U \cdot G$,	
$pr^*_U = pr'_U \oplus H(ID'_U \|\| Pwd'_U \|\| \sigma_U)$,	
$\beta^*_U = \beta_U \oplus H(r'_1 \|\| \sigma_U \|\| Pwd'_U)$,	
$r^*_1 = r'_1 \oplus H(Pwd'_U \|\| \sigma_U \|\| ID'_U)$,	
$Ver_U = H(r'_1 \|\| Pwd'_U \|\| pr_U \|\| \beta_U$	
$\|\| \sigma_U \|\| ID'_U \|\| \tau_U)$.	
Store credentials	
$\{Enc_{H(Pwd'_U \|\| ID'_U \|\| \sigma_U)}[(TID'_U, RID'_U), pr^*_U, \beta^*_U, r^*_1]$,	
Ver_U, τ_U, $H(\cdot)$, $E_q(a,b)$, G, $Gen(\cdot)$, $Rep(\cdot)$, $et\}$.	
Publish Pub'_U as the public key.	

Figure 4.6: Summary of user mobile device revocation phase

4.3.5 Dynamic IoT smart device addition phase

The IoT smart sensor devices deployed in the agricultural field work on limited resources and are susceptible to exhaustion of power, damage from surroundings, and physical capture

by an unwanted, unknown party. Such devices are rendered useless and need to be replaced with new devices. In this phase, the replacement of an old device with a new device, say SN^{new} is taken care of the following steps:

- *Step* DSA_1: TRA picks a real identity ID_S^{new}, temporary identity TID_S^{new}, and a random secret $s_1^{new} \in Z_q^*$ for SN^{new}. TRA also computes a pseudo-identity for SN^{new} as $RID_S^{new} = H(ID_S^{new} || s_1^{new})$.

- *Step* DSA_2: TRA picks a random private key $pr_S^{new} \in Z_q^*$ and computes the corresponding public key $Pub_S^{new} = pr_S^{new} \cdot G$ for SN^{new}. TRA computes a temporal credential for SN^{new} as $TC_S^{new} = H(RID_S^{new} || pr_S^{new} || pr_{TRA} || RTS_S^{new})$, where RTS_S^{new} is the current timestamp of registration of SN^{new}.

- *Step* DSA_3: TRA preloads SN^{new} with $\{(RID_S^{new}, TID_S^{new}, TC_S^{new}), H(\cdot), E_q(a,b), G, (pr_S^{new}, Pub_S^{new})\}$. In addition, TRA publishes Pub_S^{new} as SN^{new}'s public key.

The dynamic IoT smart device addition phase is briefed in Figure 4.7.

Trusted Registration Authority (TRA)	IoT Smart Device (SN)
Pick ID_S^{new}, TID_S^{new}, $s_1^{new} \in Z_q^*$.	
Compute $RID_S^{new} = H(ID_S^{new} \|\| s_1^{new})$.	
Pick random private key $pr_S^{new} \in Z_q^*$.	
Calculate $Pub_S^{new} = pr_S^{new} \cdot G$,	
Compute $TC_S^{new} = H(RID_S^{new} \|\| pr_S^{new} \|\| pr_{TRA} \|\| RTS_S^{new})$	
	Preload $\{(RID_S^{new}, TID_S^{new}, TC_S^{new}), H(\cdot), E_q(a,b), G, (pr_S^{new}, Pub_S^{new})\}$ into SN^{new}'s memory.
Publish Pub_S^{new} as SN^{new}'s public key.	

Figure 4.7: Summary of dynamic IoT smart device addition phase

4.3.6 User biometric and password change phase

A registered user UD with his/her mobile device MD_U may opt to change his/her credentials at any point without the help of the TRA. To allow this updation, the following steps are undertaken:

- *Step* BPC_1: The user UD first inputs his/her identity ID_U and password Pwd_U, and then imprints the current biometrics Bio_U at the sensor of his/her mobile device MD_U.

- *Step* BPC_2: MD_U retrieves the biometric secret key σ_U by applying the fuzzy extractor reproduction function as $Rep(Bio_U, \tau_U) = \sigma_U$. MD_U then calculates $H(Pwd_U \,||ID_U \,||\sigma_U)$ to decrypt and retrieve $\{(TID_U, RID_U), pr_U^*, \beta_U^*, r_1^*\} = Dec_{H(Pwd_U||ID_U||\sigma_U)}$ $[Enc_{H(Pwd_U||ID_U||\sigma_U)}\ [(TID_U, RID_U), pr_U^*, \beta_U^*, r_1^*]]$. MD_U then calculates $r_1 = r_1^* \oplus H(Pwd_U||\ \sigma_U||ID_U)$, $\beta_U = \beta_U^* \oplus H(r_1||\ \sigma_U||\ Pwd_U)$ and $pr_U = pr_U^* \oplus H(ID_U||\ Pwd_U||\ \sigma_U)$. It also re-computes the verifier $Ver'_U = H(r_1 \,||Pwd_U \,||pr_U \,||\beta_U \,||\sigma_U \,||ID_U \,||\tau_U)$ and checks $Ver'_U \stackrel{?}{=} Ver_U$. If the check is valid, UD is a genuine user and his/her identity ID_U, password Pwd_U, and biometric template Bio_U are legitimate. Otherwise, this phase is immediately terminated by the user UD.

- *Step* BPC_3: MD_U now prompts the user UD to enter his/her new password and biometrics. UD inputs new password Pwd_U^{new} and also imprints new biometric template Bio_U^{new} at the mobile device MD_U. Note that since the user biometric does not change over the time and if the user UD wants to keep the same biometrics Bio_U, in that case Bio_U^{new} will be treated as Bio_U.

- *Step* BPC_4: MD_U computes $RPW_U^{new} = H(Pwd_U^{new}||r_1)$ and $\beta'_U = (\beta_U \oplus H(Pwd_U \,||r_1)) \oplus RPW_U^{new} = H(RID_U \,||pr_{TRA} \,||RTS_U) \oplus RPW_U^{new}$, and then generates new biometric secret key σ_U^{new} and public reproduction parameter τ_U^{new} as $Gen(Bio_U^{new}) = (\sigma_U^{new}, \tau_U^{new})$. Furthermore , UD computes $pr_U^{new} = pr_U \oplus H(ID_U||\ Pwd_U^{new}||\ \sigma_U^{new})$, $\beta_U^{new} = \beta'_U \oplus H(r_1||\ \sigma_U^{new}||\ Pwd_U^{new})$ and $r_1^{new} = r_1 \oplus H(Pwd_U^{new}||\ \sigma_U^{new}||ID_U)$.

- *Step* BPC_5: UD also calculates a verifier $Ver_U^{new} = H(r_1 \,||Pwd_U^{new} \,||pr_U \,||\beta'_U \,||\sigma_U^{new} \,||ID_U \,||\tau_U^{new})$ and encrypts $\{(TID_U, RID_U), pr_U^{new}, \beta_U^{new}, r_1^{new}\}$ with $H(Pwd_U^{new} \,||ID_U \,||\sigma_U^{new})$ as the key. MD_U then stores the credentials $\{Enc_{H(Pwd_U^{new}||ID_U||\sigma_U^{new})}[(TID_U, RID_U), pr_U^{new}, \beta_U^{new}, r_1^{new}], Ver_U^{new}, \tau_U^{new}, H(\cdot), E_q(a,b), G, Gen(\cdot), Rep(\cdot), et\}$.

This phase is also briefed in Figure 4.8.

The power consumption of an IoT network involves energy expended to run the rechargeable devices, continuously powered devices, servers and routers that enable connectivity, and the energy expended for authenticated security. The energy expended for the authentication scheme can be calculated as the energy expended for computation of cryptographic operations and communication of messages. The proposed user authentication protocol is based on

User (UD)	Mobile device MD_U
Input identity ID_U and password Pwd_U.	
Imprint current biometrics Bio_U.	
	Retrieve $Rep(Bio_U, \tau_U) = \sigma_U$.
	Compute $H(Pwd_U \|\|ID_U \|\|\sigma_U)$.
	Retrieve $\{(TID_U, RID_U), pr_U^*, \beta_U^*, r_1^*\}$
	$= Dec_{H(Pwd_U\|\|ID_U\|\|\sigma_U)}$ $[Enc_{H(Pwd_U\|\|ID_U\|\|\sigma_U)}$
	$[(TID_U, RID_U), pr_U^*, \beta_U^*, r_1^*]]$.
	Calculate $r_1 = r_1^* \oplus H(Pwd_U\|\| \sigma_U\|\|ID_U)$,
	$\beta_U = \beta_U^* \oplus H(r_1\|\| \sigma_U\|\| Pwd_U)$,
	$pr_U = pr_U^* \oplus H(ID_U\|\| Pwd_U\|\| \sigma_U)$,
	$Ver'_U = H(r_1 \|\|Pwd_U \|\|pr_U \|\|\beta_U \|\|\sigma_U \|\|ID_U \|\|\tau_U)$.
	Check $Ver'_U \stackrel{?}{=} Ver_U$.
	If valid, prompt UD to enter his/her
	new password and biometrics.
Input new password Pwd_U^{new}.	
Imprint new biometric template Bio_U^{new}.	
	Compute $RPW_U^{new} = H(Pwd_U^{new}\|\|r_1)$,
	$\beta'_U = (\beta_U \oplus H(Pwd_U \|\|r_1)) \oplus RPW_U^{new}$
	$= H(RID_U \|\|pr_{TRA} \|\|RTS_U) \oplus RPW_U^{new}$,
	$Gen(Bio_U^{new}) = (\sigma_U^{new}, \tau_U^{new})$,
	$pr_U^{new} = pr_U \oplus H(ID_U\|\| Pwd_U^{new}\|\| \sigma_U^{new})$,
	$\beta_U^{new} = \beta'_U \oplus H(r_1\|\| \sigma_U^{new}\|\| Pwd_U^{new})$,
	$r_1^{new} = r_1 \oplus H(Pwd_U^{new}\|\| \sigma_U^{new}\|\|ID_U)$,
	$Ver_U^{new} = H(r_1 \|\|Pwd_U^{new} \|\|pr_U \|\|\beta'_U$
	$\|\|\sigma_U^{new} \|\|ID_U \|\|\tau_U^{new})$.
	Encrypt $\{(TID_U, RID_U), pr_U^{new}, \beta_U^{new}, r_1^{new}\}$
	with $H(Pwd_U^{new} \|\|ID_U \|\|\sigma_U^{new})$ as the key.
	Store $\{Enc_{H(Pwd_U^{new}\|\|ID_U\|\|\sigma_U^{new})}[(TID_U, RID_U),$
	$pr_U^{new}, \beta_U^{new}, r_1^{new}], Ver_U^{new}, \tau_U^{new}, H(\cdot), E_q(a,b),$
	$G, Gen(\cdot), Rep(\cdot), et\}$ in its memory.

Figure 4.8: Summary of user biometric and password change phase

elliptic curve cryptography. According to Potlapally *et al.* [257, 258] , ECDSA utilizes comparably low energy of 226.65 mJ for generation of 163-bit key and very low energy of 134.20 mJ and 196.23 mJ for signature generation and verification, respectively, compared to RSA

and DSA with 1024 bit-keys. Thus, authentication takes only very slight power consumption in exchange for critical security features such as anonymity, untraceability, dynamic node addition, and resistance to ESL attacks, privileged insider attacks, replay attacks, MiTM attacks, DoS attacks, offline guessing attacks. The advanced security features are provided as a result of the scheme design using ECC and hash functions. Thus, the power consumption of the advanced security features is the same as the power consumed by the elliptic curve operations and hashing algorithms.

Table 4.2: Energy costs of digital signature algorithms [257, 258]

Algorithm	Key Size (bits)	Sign (mJ)	Verify (mJ)	Key generation (mJ)
ECDSA	409	611.40	895.98	1034.92
ECDSA	283	298.86	437.00	504.96
ECDSA	233	191.37	279.82	323.30
ECDSA	193	166.75	243.84	281.65
ECDSA	163	134.20	196.23	226.65
DSA	1024	313.60	338.02	293.20
RSA	1024	546.50	15.97	270.13

Table 4.3: Energy costs of hash algorithms [257, 258]

Algorithm	SHA	SHA-1	MD5	HMAC
Energy (μJ/B)	0.75	0.76	0.59	1.16

Recently, research on optimizing Secure Hash Algorithm (SHA-256) algorithm has gained attention due to its widepread adoption for IoT and blockchain applications. SHA-256 can be optimized for blockchain-based IoT application using hardware accelerators with loop unrolling and reordering of registers to reach optimal power efficiency on both pipelined and non-pipelined configurations [230]. Tran *et al.* [314] increase the processing rate of SHA-256 by reducing the critical path delay using a multimemory processing element, a pipelined arithmetic logic unit and shift in shift out buffer.

4.4 Security analysis

We first prove the correctness of the proposed scheme for the login and authentication phase described in Section 4.3.3. Next, through both the formal security analysis under the random oracle model, namely the widely-applied "Real-Or-Random (ROR) model" [43] and non-mathematical (informal) security analysis, we show the proposed scheme is robust against several potential attacks that are needed for an IoT-enabled intelligent precision agricultural environment. In addition, in order to prove the security of the proposed scheme, the formal security verification using the broadly-accepted "Automated Validation of Internet Security Protocols and Applications (AVISPA) tool" [9] has been provided in Section 4.5.

4.4.1 Correctness proof

In Theorem 4.1, we prove that the session keys computed at the user UD and IoT smart device SN are correct and same.

Theorem 4.1. *During the "login and authentication phase described in Section 4.3.3", the session key SK_{US} computed by a user UD and the session key SK_{SU} computed by an IoT smart device SN are same.*

Proof. During the login and authentication phase of the proposed scheme, the user UD computes the shared session key with the accessed IoT smart device SN as follows:

$$\begin{aligned} SK_{SU} &= H(H(r_3||RID_S||RID_U||TC_S||T_2)||H(r_4||RID_S||TC_S||T_3) \\ &\quad ||H(y_3||RID_S||pr_S) \cdot Y_1). \end{aligned} \tag{4.1}$$

On the other side, SN also computes that shared session key with UD as follows:

$$\begin{aligned} SK_{US} &= H(H(r_3||RID_S||RID_U||TC_S||T_2)||H(r_4||RID_S||TC_S||T_3) \\ &\quad ||H(y_1||\sigma_U||RID_U||Pwd_U) \cdot Y_3). \end{aligned} \tag{4.2}$$

Now, to show $SK_{SU} = SK_{US}$, it suffices to prove from Eqs. (4.1) and (4.2) that $H(y_3||\ RID_S||\ pr_S) \cdot Y_1 = H(y_1||\ \sigma_U\ ||RID_U\ ||Pwd_U) \cdot Y_3$. We have:

$$\begin{aligned} H(y_3||RID_S||pr_S) \cdot Y_1 &= (H(y_3||RID_S||pr_S) * H(y_1||\sigma_U||RID_U||Pwd_U)) \cdot G \\ &= H(y_1||\sigma_U||RID_U||Pwd_U) \cdot (H(y_3||RID_S||pr_S) \cdot G) \\ &= H(y_1||\sigma_U||RID_U||Pwd_U) \cdot Y_3. \end{aligned}$$

Hence, the theorem follows. □

4.4.2 Formal security analysis using ROR model

In this section, we first discuss in brief, the "Real-Or-Random (ROR) model" [43] and various random oracles associated with the model. Next, we define some computational problems and "collision-resistant one-way hash function" before proving the security of the common session key computed in the proposed scheme during the login and authentication phase described in Section 4.3.3.

Typically, the users select their passwords from a uniformly distributed passwords dictionary. However, Wang *et al.* [326] observed that by Zipf's law, actual user-taken passwords significantly diverge from the uniform distribution expected in passwords to be chosen by the user. In practice, the size of the user-chosen passwords dictionary is restricted because the entire space of the passwords may not be used by the users. Therefore, only a little part of the allowable character space is utilized [326]. In this chapter, we also use Zipf's law for guessing the user-chosen passwords by an adversary, say $\mathcal{A}$.

In the following, we discuss briefly the basics of the ROR model prior to proving the all-round security of session key computed in the proposed scheme in Theorem 4.2.

- **Participants.** In the proposed scheme, the four participants, namely the TRA, the user UD, the controller node CN and the IoT smart device SN are involved during the enrollment and the login and authentication phases. The instances l_1, l_2, l_3 and l_4 of TRA, UD, CN and SN are used to create the representations of the "random oracles" by the symbols, say $\Pi_{TRA}^{l_1}$, $\Pi_{UD}^{l_2}$, $\Pi_{CN}^{l_3}$ and $\Pi_{SN}^{l_4}$, for the four participants respectively.

- **Accepted state.** An instance Π^{l} will be in an "accepted state", if it goes to an accept state after receiving the last authentic protocol message. If we arrange all the sent and received messages in sequence, it forms the "session identification sid of Π^{l} for the current session".

- **Partnering.** Two instances, say Π^{l_1} and Π^{l_2} will be called partners to each other, if the following three criteria are satisfied simultaneously:

  - Both instances are in "accepted states".
  - Both instances will have the same sid and they need to "mutually authenticate each other".
  - Both instances are "mutual partners of each other".

- **Freshness.** An instance $\Pi_{UD}^{l_2}$ or $\Pi_{SN}^{l_4}$ is said to be fresh if "the generated session key SK_{US} $(= SK_{SU})$ between the user UD and the smart device SN is not leaked to the adversary $\mathcal{A}$ by having the Reveal(Π^{l_i}) query listed in Table 4.4.

Various queries that are accessible by $\mathcal{A}$ are also listed in Table 4.4.

Table 4.4: Queries and their purposes

Query	Purpose
$Send(\Pi^l, msg)$	Using such a query $\mathcal{A}$ can send a message msg to an instance Π^l, and Π^l may respond accordingly for the received message msg
$Execute(\Pi_{UD}^{l_2}, \Pi_{CN}^{l_2}, \Pi_{SN}^{l_4})$	This query helps $\mathcal{A}$ to intercept the messages communicated among the participants UD, CN and SN
$CorruptMD(\Pi_{UD}^{l_2})$	This query aids $\mathcal{A}$ to procure the secret credentials kept in MD_U from the "user UD's lost or stolen mobile device MD_U"
$CorruptSN(\Pi_{SN}^{l_4})$	This query helps $\mathcal{A}$ to access the secret credentials stored in a compromised IoT smart device SN.
$Reveal(\Pi^l)$	The session key SK_{US} $(= SK_{SU})$ between Π^l and its corresponding partner in a present session is revealed to the adversary $\mathcal{A}$
$Test(\Pi^l)$	With the help of this query, $\mathcal{A}$ can check the validity of the derived session key SK_{US} and Π^l will provide with a "probabilistic outcome of a flipped unbiased coin, say c"

We now define the semantic security that gives the session key security for attempting the session key SK_{US} $(= SK_{SU})$ between UD and SN in between the communication.

Definition 4.1 (Semantic security). *If $Adv_{\mathcal{A}}^{AKS}(t_p)$ represents the "advantage (success probability) of an adversary $\mathcal{A}$ running in polynomial time t_p to derive the session key SK_{US} $(= SK_{SU})$ between a user UD and an IoT smart device SN in the proposed authenticated key agreement scheme (AKS), $Adv_{\mathcal{A}}^{AKS}(t_p) = |2Pr[c' = c] - 1|$, where c and c' denote the "correct" and "guessed" bits, and $Pr[X]$ is the "probability of a random event X".*

We also define the collision-resistant one-way hash function and the computational problem of "elliptic curve decisional Diffie-Hellman problem (ECDDHP)" as follows.

Definition 4.2 (One-way collision-resistant hash function)**.** *A "one-way collision-resistant hash function", say H: $\{0,1\}^* \rightarrow \{0,1\}^{l_h}$ is a "deterministic algorithm which gives output as a binary string $H(s) \in \{0,1\}^{l_h}$ of fixed-length l_h bits as hash output (message digest) on an input with an arbitrary length binary string $s \in \{0,1\}^*$". The advantage in finding collision for an adversary $\mathcal{A}$ is then "$Adv_{\mathcal{A}}^{HASH}(t_p) = Pr[(sr_1, sr_2) \leftarrow_R \mathcal{A} : sr_1 \neq sr_2,\ H(sr_1) = H(sr_2)]$, where $(sr_1, sr_2) \leftarrow_R \mathcal{A}$ indicates that the input pair (sr_1, sr_2) is randomly picked by $\mathcal{A}$". An (ψ, t_p)-adversary $\mathcal{A}$ that attacks H's collision resistance suggests that the run-time that is allowed for $\mathcal{A}$ cannot exceed t_p and $Adv_{\mathcal{A}}^{HASH}(t_p) \leq \psi$.*

Definition 4.3 (Elliptic curve decisional Diffie-Hellman problem (ECDDHP))**.** *Let $P \in E_q(a,b)$ be an elliptic curve point. Given a quadruple (P, $k_1.P$, $k_2.P$, $k_3.P$), decide "whether $k_3 = k_1 * k_2$ or it is a uniform value, where k_1, k_2, $k_3 \in Z_q^*$".*

In order to maintain the intractability of ECDDHP, the prime q must be picked as at least 160-bit number.

Theorem 4.2. *Consider an adversary $\mathcal{A}$ to be a polynomial time adversary that runs in time t_p against the proposed scheme (AKS) under the ROR model. If the Zipf's law is applied on the user-selected passwords, l_b is the number of bits present in the biometrics secret key σ_U, and $Adv_{\mathcal{A}}^{AKS}(t_p)$ is the $\mathcal{A}$'s advantage in breaking the proposed scheme's semantic security in time t_p for deriving the session key SK_{US} $(= SK_{SU})$ between the user UD and the IoT smart device SN (see Definition 4.1), then*

$$Adv_{\mathcal{A}}^{AKS}(t_p) \leq \frac{q_{hsh}^2}{|Hash|} + 2\big[\max\{C'.q_{snd}^{s'}, \frac{q_{snd}}{2^{l_b}}\} + Adv_{\mathcal{A}}^{ECDDHP}(t_p)\big],$$

where q_{hsh}, q_{snd} and $|Hash|$ are the "number of hash queries, Send queries and range space of $H(\cdot)$", respectively, and $Adv_{\mathcal{A}}^{ECDDHP}(t_p)$ is the $\mathcal{A}$'s advantage in breaking the ECDDHP (see Definition 4.3), and C' and s' are the Zipf's parameters [326].

Proof. We define four games, say $Gm_j^{\mathcal{A}}$, $j = 0, 1, 2, 3$ that are associated with the adversary $\mathcal{A}$ in the proposed authenticated key agreement scheme (AKS). Let $Succ_{Gm_j}^{\mathcal{A}}$ denote an event where $\mathcal{A}$ can guess the "random bit c in the game $Gm_j^{\mathcal{A}}$ correctly", and $\mathcal{A}$'s advantage in winning $Gm_j^{\mathcal{A}}$ is defined by $Adv_{Gm_j^{\mathcal{A}}} = Pr[Succ_{Gm_j}^{\mathcal{A}}]$.

The detailed description of each game is outlined below:

- $Gm_0^{\mathcal{A}}$: This game represents the real attack performed by the adversary $\mathcal{A}$ against our proposed scheme (AKS) in the ROR model. Prior to beginning of the game $Gm_0^{\mathcal{A}}$, $\mathcal{A}$

needs to pick the bit c randomly. From the semantic security of the proposed scheme (AKS) defined in Definition 4.1, we have:

$$Adv_{\mathcal{A}}^{AKS}(t_p) = |2Adv_{Gm_0^{\mathcal{A}}} - 1| \tag{4.3}$$

- $Gm_1^{\mathcal{A}}$: This game implements an eavesdropping attack, where the adversary $\mathcal{A}$ makes use of the $Execute(\Pi_{UD}^{l_2}, \Pi_{CN}^{l_2}, \Pi_{SN}^{l_4})$ query to intercept the messages $Msg_1 = \langle TID_U, M_1, Y_1, Sign_{y_1}, T_1\rangle$, $Msg_2 = \langle TID_C, M_2, Y_1, Y_2, Sign_{y_2}, T_2\rangle$, $Msg_3 = \langle TID_S^*, M_3, M_4, Y_3, Sign_{y_3}, T_3\rangle$ and $Msg_4 = \langle TID_U^*, M_3, M_4, M_5, Y_3, Y_4, Sign_{y_4}, T_3, T_4\rangle$ during the login and authentication phase of the proposed scheme. After that $\mathcal{A}$ may use the information obtained from the intercepted messages to derive the session key SK_{US} $(= SK_{SU})$, where $SK_{SU} = H(H(r_3||\ RID_S||\ RID_U||\ TC_S||\ T_2)||\ H(r_4||\ RID_S||\ TC_S||\ T_3)||\ H(y_3||\ RID_S||\ pr_S) \cdot Y_1) = H(H(r_3||\ RID_S||\ RID_U||\ TC_S||\ T_2)\ ||H(r_4||\ RID_S||\ TC_S||\ T_3)\ ||H(y_1||\ \sigma_U||\ RID_U||\ Pwd_U) \cdot Y_3) = SK_{US}$. It is clear that the session key construction is based on the temporal (short-term) secrets (r_3, r_4, y_1, y_3) and the long-term secrets (RID_S, RID_U, TC_S, pr_S, σ_U, Pwd_U). However, none of these credentials can be obtained from the intercepted messages. Now, $\mathcal{A}$ needs to execute the *Reveal* and $Test$ queries in order to check whether the derived session key is the correct one or just a random key. Since only intercepting the messages will not increase the success probability for deriving the session key SK_{US} $(= SK_{SU})$, the games $Gm_0^{\mathcal{A}}$ and $Gm_1^{\mathcal{A}}$ are indistinguishable under the eavesdropping attack. Hence, we get the following result:

$$Adv_{Gm_1^{\mathcal{A}}} = Adv_{Gm_0^{\mathcal{A}}} \tag{4.4}$$

- $Gm_2^{\mathcal{A}}$: In this game, an active attack is modelled in which the adversary $\mathcal{A}$ executes the $CorruptMD(\Pi_{UD}^{l_2})$ and $CorruptSN(\Pi_{SN}^{l_4})$ queries. Therefore, using the $CorruptMD(\Pi_{UD}^{l_2})$ query, $\mathcal{A}$ has the user credentials $\{Enc_{H(Pwd_U||ID_U||\sigma_U)}[(TID_U, RID_U), pr_U^*, \beta_U^*, r_1^*], Ver_U, \tau_U, H(\cdot), E_q(a,b), G, Gen(\cdot), Rep(\cdot), et\}$ that are available in the user UD's mobile device MD_U. On the other side, execution of $CorruptSN(\Pi_{SN}^{l_4})$ query will allow $\mathcal{A}$ to acquire the credentials $\{(RID_S, TID_S, TC_S), H(\cdot), E_q(a,b), G, (pr_S, Pub_S)\}$ stored in a compromised IoT smart device SN's memory. The secrets (RID_S, TC_S, pr_S) are not helpful in constructing the session key SK_{US} $(= SK_{SU})$ because it also requires other secret credentials (r_3, r_4, y_1, y_3, RID_U, σ_U, Pwd_U). Also, deriving/guessing (RID_U, σ_U, Pwd_U) is difficult from $Enc_{H(Pwd_U||ID_U||\sigma_U)}[(TID_U, RID_U), pr_U^*, \beta_U^*, r_1^*]$ using the *Send* query as ID_U is unavailable. The fuzzy extractor method [124] in the proposed scheme can extract at

most l_b nearly random bits. The probability of guessing σ_U can then be estimated approximately as $\frac{1}{2^{l_b}}$ [249]. The games $Gm_1^{\mathcal{A}}$ and $Gm_2^{\mathcal{A}}$ become identical in the absence of the "password/biometrics guessing attacks". The Zipf's law on user-selected passwords [326] leads to the following relation:

$$|Adv_{Gm_1^{\mathcal{A}}} - Adv_{Gm_2^{\mathcal{A}}}| \leq \max\left\{C'.q_{snd}^{s'}, \frac{q_{snd}}{2^{l_b}}\right\} \tag{4.5}$$

- $Gm_3^{\mathcal{A}}$: Under this game, an active attack is modelled in which the adversary $\mathcal{A}$ executes the simulation of hash queries $Hash$, which behaves like random oracles, and also need to solve the computational ECDDHP defined in Definition 4.3 to derive the session key SK_{US} $(= SK_{SU})$. The construction of this session key depends on both the temporal (short-term) secrets (r_3, r_4, y_1, y_3) as well as the long-term secrets (RID_S, RID_U, TC_S, pr_S, σ_U, Pwd_U). Moreover, to derive either $H(y_3||\ RID_S||\ pr_S)\cdot Y_1$ or $H(y_1||\ \sigma_U||\ RID_U||\ Pwd_U)\cdot Y_3$ from the eavesdropped Y_1 and Y_3, an adversary $\mathcal{A}$ needs to solve the ECDDHP in polynomial time t_p defined in Definition 4.3 whose advantage is $Adv_{\mathcal{A}}^{ECDDHP}(t_p)$. In addition, to obtain other secret credentials that are embedded in the hash, $\mathcal{A}$ needs to find the collision in the hash function $H(\cdot)$ using q_{hsh} number of $Hash$ queries. In the absence of the simulation of hash queries and solving ECDDHP, both the games $Gm_2^{\mathcal{A}}$ and $Gm_3^{\mathcal{A}}$ are "indistinguishable". The birthday paradox result and $Adv_{\mathcal{A}}^{ECDDHP}(t_p)$ provide the following relation:

$$|Adv_{Gm_2^{\mathcal{A}}} - Adv_{Gm_3^{\mathcal{A}}}| \leq \frac{q_{hsh}^2}{2|Hash|} + Adv_{\mathcal{A}}^{ECDDHP}(t_p) \tag{4.6}$$

Since the games are over, $\mathcal{A}$ is left only to guess the correct bit c. It is then obvious that

$$Adv_{Gm_3^{\mathcal{A}}} = \frac{1}{2} \tag{4.7}$$

Now, Eqs. (4.3), (4.4) and (4.7) give

$$\begin{aligned} \frac{1}{2}.Adv_{\mathcal{A}}^{AKS}(t_p) &= |Adv_{Gm_0^{\mathcal{A}}} - \frac{1}{2}| \\ &= |Adv_{Gm_1^{\mathcal{A}}} - Adv_{Gm_3^{\mathcal{A}}}| \end{aligned} \tag{4.8}$$

With the help of Eqs. (4.5), (4.6) and (4.8), and the triangular inequality, we have the following relation:

$$\begin{aligned} \frac{1}{2}.Adv_{\mathcal{A}}^{AKS}(t_p) &= |Adv_{Gm_1^{\mathcal{A}}} - Adv_{Gm_3^{\mathcal{A}}}| \\ &\leq |Adv_{Gm_1^{\mathcal{A}}} - Adv_{Gm_2^{\mathcal{A}}}| + |Adv_{Gm_2^{\mathcal{A}}} - Adv_{Gm_3^{\mathcal{A}}}| \\ &\leq \max\left\{C'.q_{snd}^{s'}, \frac{q_{snd}}{2^{l_b}}\right\} + \frac{q_{hsh}^2}{2|Hash|} + Adv_{\mathcal{A}}^{ECDDHP}(t_p) \end{aligned} \tag{4.9}$$

Finally, by multiplying both sides of Eq. (4.9) by a factor of 2 and rearranging the terms, we arrive to the desired result:

$$Adv_{\mathcal{A}}^{AKS}(t_p) \leq \frac{q_{hsh}^2}{|Hash|} + 2\big[\max\{C'.q_{snd}^{s'}, \frac{q_{snd}}{2^{l_b}}\} + Adv_{\mathcal{A}}^{ECDDHP}(t_p)\big].$$

□

4.4.3 Informal security analysis

Through the non-mathematical security analysis in the following propositions we show that the proposed scheme can resist against several known attacks.

Proposition 4.1. *Ephemeral secret leakage (ESL) attack is protected in the proposed scheme.*

Proof. During the login and authentication phase of the proposed scheme, the user UD computes the shared session key with the accessed IoT smart device SN as $SK_{SU} = H(H(r_3||\ RID_S||\ RID_U||\ TC_S||\ T_2)\ ||H(r_4||\ RID_S||\ TC_S||\ T_3)\ ||H(y_3||\ RID_S||\ pr_S) \cdot Y_1)$. On the other side, SN also computes that shared session key with UD as $SK_{US} = H(H(r_3||\ RID_S||\ RID_U||\ TC_S||\ T_2)\ ||H(r_4||\ RID_S||\ TC_S||\ T_3)\ ||\ H(y_1||\ \sigma_U||\ RID_U||\ Pwd_U)\cdot Y_3)$, which is same as SK_{SU} (see Theorem 4.1). To obtain either $H(y_3||\ RID_S||\ pr_S)\cdot Y_1$ or $H(y_1||\ \sigma_U||\ RID_U||\ Pwd_U)\cdot Y_3$ from the eavesdropped Y_1 and Y_3, an adversary $\mathcal{A}$ needs to solve the ECDDHP defined in Definition 4.3, which is computationally infeasible problem. It is worth noticing that the session key is dependent on both the temporal (short-term) secrets (r_3, r_4, y_1, y_3) and long-term secrets (RID_S, RID_U, TC_S, pr_S, σ_U, Pwd_U).

We now consider the following two cases:

- **Case 1.** If only the temporal (short-term) secrets (r_3, r_4, y_1, y_3) are compromised, the session key SK_{US} ($= SK_{SU}$) is not compromised without possessing the long-term secrets (RID_S, RID_U, TC_S, pr_S, σ_U, Pwd_U).
- **Case 2.** If only the long-term secrets (RID_S, RID_U, TC_S, pr_S, σ_U, Pwd_U) are compromised, the session key SK_{US} ($= SK_{SU}$) is not derived without possessing the short-term secrets (r_3, r_4, y_1, y_3).

Thus, the session is compromised when both the short and long-term secrets are compromised by the adversary $\mathcal{A}$. Based on the CK-adversary model, the proposed scheme is then resilient against ESL attack. □

Proposition 4.2. *The proposed scheme is resilient against privileged insider attack.*

Proof. The registration phase of the "smart sensor device" and the "controller node" do not require any passing of secret credentials to the "trusted registration authority". Instead, the TRA preloads the secret credentials into CN and SN. Also, during the registration of a user UD, the messages passed include only the data required to be stored at the TRA via secure channel. The creation of user secret credentials uses random secrets generated at the user side and does not involve the need for any parameters received from the TRA. It is worth noticing that the user UD sends the registration information $\{RID_U, RPW_U \oplus r_2\}$ to the TRA via secure channel, where r_1 and r_2 are random secrets only known to the user UD, and $RID_U = H(ID_U || r_1)$ and $RPW_U = H(Pwd_U || r_1)$. Therefore, a privileged-insider user of the TRA, being an insider attacker, knows the information $\{RID_U, RPW_U \oplus r_2\}$. However, without secrets r_1 and r_2, the attacker can not correctly guess identity and password of the user UD due to the collision resistant property of the one-way hash function $H(\cdot)$ (see Definition 4.2). Thus, the proposed scheme is strongly resilient against "privileged insider attack". □

Proposition 4.3. *The proposed scheme is resilient against replay attack.*

Proof. Assume that an adversary, say $\mathcal{A}$ intercepts the messages $Msg_1 = \langle TID_U, M_1, Y_1, Sign_{y_1}, T_1 \rangle$, $Msg_2 = \langle TID_C, M_2, Y_1, Y_2, Sign_{y_2}, T_2 \rangle$, $Msg_3 = \langle TID_S^*, M_3, M_4, Y_3, Sign_{y_3}, T_3 \rangle$ and $Msg_4 = \langle TID_U^*, M_3, M_4, M_5, Y_3, Y_4, Sign_{y_4}, T_3, T_4 \rangle$ during the login and authentication phase of the proposed scheme. To deter the adversary $\mathcal{A}$ from replaying the old messages, every message includes either timestamps or random secrets or both, which are verified by the receiver before any processing of the received message(s). Thus, if the receiver encounters an old timestamp or a reused random secret, the message is detected and automatically discarded. This shows that the proposed scheme protects against replay attack. □

Proposition 4.4. *The proposed scheme is resilient against man-in-the-middle attack.*

Proof. Assume that an adversary $\mathcal{A}$ intercepts the messages Msg_1, Msg_2, Msg_3 and Msg_4, and tries to tamper the contents before passing it to the intended recipient so that the recipient will not be aware of the modified messages. In our authentication scheme, in the message Msg_1, the ciphertext and verifiers M_1 and Y_1, and signature $Sign_{y_1}$ cannot be tampered as they use secret credentials r_3, y_1, pr_U, Pwd_U and σ_U that are known only to UD. Tampering of TID_U can be immediately identified at CN by verifying the signature. Similarly, the parameters M_2, Y_1, Y_2, and $Sign_{y_2}$ in Msg_2 use y_2, the parameters M_3, M_4,

Y_3, and $Sign_{y_3}$ in Msg_3 use r_4, y_3, pr_C and pr_S, and the parameters M_3, M_4, M_5, Y_3, Y_4 and $Sign_{y_4}$ in Msg_4 use y_4. Hence, $\mathcal{A}$ cannot tamper with any messages on the fly. Thus, the proposed scheme is resilient against "man-in-the-middle attack". □

Proposition 4.5. *The proposed scheme is secure against impersonation attacks.*

Proof. Suppose an adversary $\mathcal{A}$ intercepts the messages $Msg_1 = \langle TID_U, M_1, Y_1, Sign_{y_1}, T_1\rangle$, $Msg_2 = \langle TID_C, M_2, Y_1, Y_2, Sign_{y_2}, T_2\rangle$, $Msg_3 = \langle TID_S^*, M_3, M_4, Y_3, Sign_{y_3}, T_3\rangle$ and $Msg_4 = \langle TID_U^*, M_3, M_4, M_5, Y_3, Y_4, Sign_{y_4}, T_3, T_4\rangle$ during the login and authentication phase of the proposed scheme.

We consider the following three cases:

- **User impersonation attack**: In this attack, the adversary $\mathcal{A}$ tries to deceive the controller node CN by impersonating the legitimate registered user UD. To achieve this goal, $\mathcal{A}$ may generate random secrets r_3^a, $y_1^a \in Z_q^*$ and the current timestamp T_1^a, and calculate $RID_U^* = H(RID_U || T_1^a)$, $M_1^a = RID_U^* \oplus (r_3^a || RID_S)$, and $Y_1^a = H(y_1^a || \sigma_U || RID_U || Pwd_U) \cdot G$, and the signature on r_3^a and y_1^a as $Sign_{y_1}^a = H(y_1^a || \sigma_U || RID_U || Pwd_U) + H(r_3^a || M_1^a || RID_U || RID_S || T_1^a || Pub_U) * pr_U \pmod q$ in order to create a valid message $Msg_1^a = \langle TID_U, M_1^a, Y_1^a, Sign_{y_1}^a, T_1^a\rangle$ and then send to CN on behalf of UD. However, this is not possible for $\mathcal{A}$ without having the knowledge of the long-term secrets (RID_U, RID_S, Pwd_U, σ_U, pr_U). Thus, user impersonation attack is resisted in the proposed scheme.

- **Controller node impersonation attack**: In this attack, assume that $\mathcal{A}$ tries to deceive the smart device SN by impersonating the controller node CN by sending a created message, say $Msg_2^a = \langle TID_C, M_2^a, Y_1^a, Y_2^a, Sign_{y_2}^a, T_2^a\rangle$ and also to deceive the user UD by impersonating the controller node CN by sending a generated message, say $Msg_4^a = \langle TID_U^*, M_3^a, M_4^a, M_5^a, Y_3^a, Y_4^a, Sign_{y_4}^a, T_3^a, T_4^a\rangle$. Suppose $\mathcal{A}$ generates timestamps T_2^a, T_3^a and T_4^a, and random secrets $y_2^a, y_4^a \in Z_q^*$, $r_3^a, r_4^a \in Z_q^*$. However, to compute $M_2^a = H(r_3^a || RID_S || RID_U || TC_S || T_2^a) \oplus H(RID_S || TC_S || T_2^a)$, $Y_2^a = H(y_2^a || RID_C || pr_C) \cdot G$ and the signature on y_2^a as $Sign_{y_2}^a = H(y_2^a || RID_C || pr_C) + H(M_2^a || RID_S || TID_C || T_2^a || Pub_C || Y_1^a) * pr_C \pmod q$, $\mathcal{A}$ must know the secrets RID_C, RID_S, TC_S, RID_U and pr_C. As a result, $\mathcal{A}$ can not send the fabricated message $Msg_2^a = \langle TID_C, M_2^a, Y_1^a, Y_2^a, Sign_{y_2}^a, T_2^a\rangle$ to SN on behalf of the CN. In a similar way, to fabricate the message $Msg_4^a = \langle TID_U^*, M_3^a, M_4^a, M_5^a, Y_3^a, Y_4^a, Sign_{y_4}^a, T_3^a, T_4^a\rangle$, the adversary $\mathcal{A}$ needs to compute $Y_4^a = H(y_4^a || RID_C || pr_C) \cdot G$ and $M_5 = H(r_3^a || RID_S ||$

$RID_U||\ TC_S||\ T_2^a) \oplus H(H(y_4^a||RID_C||pr_C) \cdot Y_1^a||\ RID_U||\ T_4^a)$ and the signature on y_4^a as $Sign_{y_4}^a = H(y_4^a||\ RID_C||\ pr_C) + H(M_3^a||\ M_4^a||M_5^a||\ Y_1^a\ ||Y_3^a||RID_U||\ T_3^a||\ T_4^a||Pub_C) * pr_C \pmod q$, where $M_3^a = H(r_4^a||\ RID_S||\ TC_S||\ T_3^a) \oplus H(H(y_3^a||\ RID_S||\ pr_S) \cdot Y_1^a||\ RID_S||\ T_3^a)$, $Y_3^a = H(y_3^a||\ RID_S||\ pr_S) \cdot G$. The construction of Msg_4^a also needs the secrets. Hence, the proposed scheme is resilient against controller node impersonation attacks.

- **IoT smart device impersonation attack**: To impersonate the smart device SN and deceive the controller node CN, the adversary $\mathcal{A}$ needs to create a valid message, say $Msg_3^a = \langle TID_S^*, M_3^a, M_4^a, Y_3^a, Sign_{y_3}^a, T_3^a \rangle$. Now, to calculate $M_3^a = H(r_4^a||\ RID_S||\ TC_S||\ T_3^a) \oplus H(H(y_3^a||\ RID_S||\ pr_S) \cdot Y_1^a||\ RID_S||\ T_3^a)$, $Y_3^a = H(y_3^a||\ RID_S||\ pr_S) \cdot G$, the common session key shared with the user UD as $SK_{SU}^a = H(H(r_3^a||\ RID_S||\ RID_U||\ TC_S||\ T_2^a)\ ||H(r_4^a||\ RID_S||\ TC_S||\ T_3^a)\ ||H(y_3^a||\ RID_S||\ pr_S) \cdot Y_1^a)$ along with its verifier $M_4^a = H(SK_{SU}^a||\ T_3^a)$ and the signature on y_3^a and r_4^a as $Sign_{y_3}^a = H(y_3^a||\ RID_S||\ pr_S) + H(M_3^a||M_4^a||\ RID_S||\ TID_C||\ T_3^a||Pub_S) * pr_S \pmod q$, $\mathcal{A}$ requires the secrets RID_C, RID_S, TC_S, RID_U and pr_S. Since the secrets are unknown to $\mathcal{A}$, IoT smart device impersonation attack is also protected against in the proposed scheme.

□

Proposition 4.6. *The proposed scheme is secure against physical smart device capture attack.*

Proof. If an adversary A physically captures a smart device SN due to unattended environment of IoT-enabled intelligent precision agricultural environment as discussed in the threat model in Section 4.1.2, he/she can extract all its credentials using the "power analysis attacks" [200]. This risks exposure of the information $\{(RID_S, TID_S, TC_S), H(\cdot), E_q(a,b), G, (pr_S, Pub_S)\}$ stored in SN along with its sensed data. The leakage of the information cannot endanger the secure communication between other non-compromised smart devices SN and a user UD as every smart device has unique credentials different from other smart devices deployed in the network. Moreover, the session keys established among the users and all the accessed smart devices are also distinct. Hence, the proposed scheme is secure against "physical device capture attack". □

Proposition 4.7. *The proposed scheme is resilient against denial-of-service (DoS) attack.*

Proof. Chang and Nguyen [98] observed that traditional one-way hash functions, biohashing and perceptual hashing techniques may reduce output error, but they hardly produce a

unique value from the biometric template of a user UD at different input times. This leads to a high rate of false rejection. To avoid such a "high rate of false rejection", we have used the fuzzy extractor method [124] because it is verified using the Hamming distance in order to avoid "false acceptance and false rejection errors" [98]. Moreover, due to the use of current timestamps in each message, even if an adversary intentionally attempts to send the same message many times, these are easily detected by checking the timestamp at the receiver end (as discussed in Proposition 4.3), and the messages are not processed further. This means that the adversary will not be able to force the entities (users, controller nodes and smart devices) to consume resources (power, storage and computation). As a result, the proposed scheme is resilient against DoS attacks. □

Proposition 4.8. *Stolen user mobile device capture attack is protected in the proposed scheme.*

Proof. Suppose the mobile device MD_U of a registered user UD has been stolen or lost, and an adversary having the mobile device MD_U can extract all the credentials $\{Enc_{H(Pwd_U||ID_U||\sigma_U)}[(TID_U,\ RID_U),\ pr_U^*,\ \beta_U^*,\ r_1^*],\ Ver_U,\ \tau_U,\ H(\cdot),\ E_q(a,b),\ G,\ Gen(\cdot),\ Rep(\cdot),\ et\}$ from its memory using the "power analysis attacks" [200]. All the credentials stored in the mobile device are encrypted using a symmetric key created using only the private credentials of the user UD, which is $H(Pwd_U\ ||ID_U\ ||\sigma_U)$. The user private credentials are unknown to the adversary because these are not stored in plaintext form anywhere. Thus, the adversary cannot gain any information by capturing the user mobile device MD_U because he/she needs to guess the identity ID_U, password Pwd_U and biometric secret key σ_U at the same time through the "offline guessing attacks". However, the offline identity, password and biometric guessing attacks are computationally infeasible due to long 160-bit values and uniqueness of ID_U and σ_U that is derived from the user's biometric template Bio_U. As a result, the proposed scheme is secure against the lost/stolen user mobile device capture attack. □

Proposition 4.9. *The proposed scheme achieves both anonymity and untraceability.*

Proof. The messages communicated among a user UD, the controller node CN and an IoT smart device SN during the login and authentication phases are $Msg_1 = \langle TID_U,\ M_1,\ Y_1,\ Sign_{y_1},\ T_1\rangle$, $Msg_2 = \langle TID_C,\ M_2,\ Y_1,\ Y_2,\ Sign_{y_2},\ T_2\rangle$, $Msg_3 = \langle TID_S^*,\ M_3,\ M_4,\ Y_3,\ Sign_{y_3},\ T_3\rangle$ and $Msg_4 = \langle TID_U^*,\ M_3,\ M_4,\ M_5,\ Y_3,\ Y_4,\ Sign_{y_4},\ T_3,\ T_4\rangle$. It is worth noticing that none of the messages include the real/pseudo identities of UD, CN and SN. In addition, all

the components of the messages are unique and random because these are distinct in each session. Thus, an adversary can not identify the communicating entities and also can not trace who is communicating with whom over successive sessions. Therefore, both anonymity and untraceability properties are preserved in the proposed scheme. □

4.5 Formal security verification using AVISPA: simulation study

An automated validation based software model-checking tool, known as "Automated Validation of Internet Security Protocols and Applications(AVISPA)" [9], is used to simulate the security protocols and conclude whether they are secure or insecure against passive/active attacks, such as "replay" and "'man-in-the-middle" attacks. To simulate a tested protocol, it is first written as "High Level Protocol Specification Language (HLPSL)" code, and then passed to a translator followed by one of the available backends in AVISPA. It employs temporal logic to imitate the security model in HLPSL consisting of the basic roles and composite roles. The basic roles allow simulation of the entities involved in the protocol and are user defined. Composite roles are sessions, goals and environments, that combine and manage the basic roles. Every HLPSL code must mandatorily have the composite roles. HLPSL allows to program the protocols as messages exchanged between different roles, with each role comprising of transitions between multiple states. The code in HLPSL language is converted into an Intermediate Format (IF) using HLPSL2IF translator, which is then fed into a backend. The architecture and working of AVISPA has been discussed in detail in Appendix A.

The proposed scheme has been implemented in HLPSL with the roles for TRA, CN, UD and SN as the basic roles. The parameters and signatures are created as defined in the scheme and messages are sent between the roles. A request-witness on the timestamps and random secrets with strong authentication goal ensures they are received correctly on the receiver side. An authentication goal is specified on all random secrets r_3, r_4, y_1, y_2, y_3 and y_4. It is assumed that the intruder (always denoted by i in HLPSL) is aware of all the timestamps used. The simulation was performed in "SPAN, the Security Protocol ANimator for AVISPA" [22]. AVISPA assures protection against "replay attack" and "man-in-the-middle attack" by employing the DY threat model [125] against a security protocol under test. The proposed scheme is verified in three ways: a) "executability checking on non-

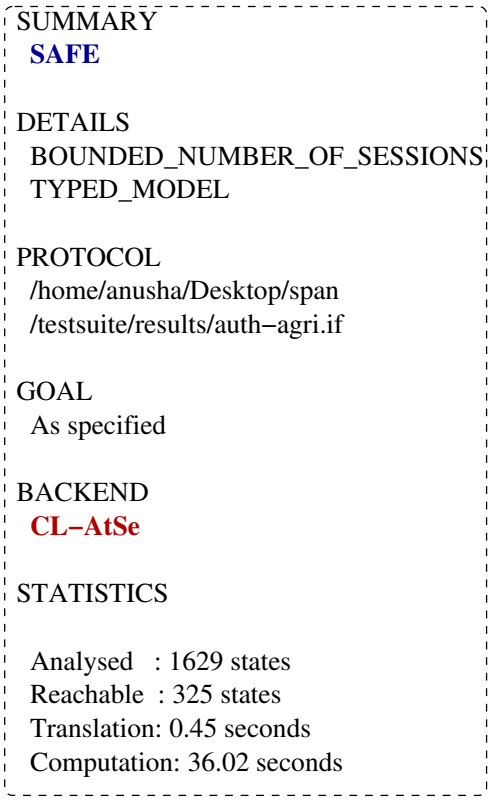

Figure 4.9: Analysis of simulation results under CL-AtSe backend

trivial HLPSL specifications", b) "replay attack checking", and c) "DY model checking". If a protocol model does not complete its execution due to any modeling mistakes, no attack may be found by the back-ends as the attack state is never reached. In such cases, "executability check for non-trivial HLPSL specifications" is used as important criteria for formal security verification under the AVISPA tool [322]. The backends search for a passive intruder and feed it with knowledge from honest agents to test against replay attack [9]. The DY model is checked by identifying if a "man-in-the-middle attack" can be launched with the identified intruder. The output results reported in Figure 4.9 show that the proposed scheme is "safe" with the CL-AtSe backend with the total number of states analysed and reached and the time for translation the protocol into constraints and the total time for computation.

4.6 Performance comparison

In this section, we provide a detailed comparative analysis on various "security and functionality features", "communication costs" and "computational costs" among the proposed

scheme and other related existing competing authentication schemes, such as the schemes of Ali *et al.* [52], Dhillon and Kalra [122], Sadhukhan *et al.* [274] and Shuai *et al.* [286].

4.6.1 Comparison of security and functionality features

Table 4.5 shows the comparative analysis on various "security and functionality features" of the proposed scheme with the schemes given in Ali *et al.* [52], Dhillon and Kalra [122], Sadhukhan *et al.* [274] and Shuai *et al.* [286]. It shows that the proposed scheme provides superior security and more functionality features as compared to all compared authentication schemes. It is worth noticing that only the proposed scheme resists the ephemeral secret leakage (ESL) attack which is a very important attack considered for the session key security in an authentication scheme. Moreover, user, controller node and IoT smart device anonymity and untraceability properties are satisfied at the same time in the proposed scheme.

4.6.2 Comparison of computation costs

In this section, we use the testbed experiments of various cryptographic primitives for server and Raspberry PI 3 settings as reported in Appendix B. If T_{fe} denotes the execution time needed for a fuzzy extractor function $(Gen(\cdot)/Rep(\cdot))$, we assume that $T_{fe} \approx T_{ecm}$ [166]. We use the average time for cryptographic primitives for a server setting reported in Table B.1 for calculating the computational costs for a controller node (server) in the schemes. On the other side, we use the average time for cryptographic primitives for a Raspberry PI 3 setting tabulated in Table B.2 for calculating the computational costs for the resource constrained device, such as IoT smart device/sensor node and user mobile device in the schemes. In the proposed scheme, during the login and authentication phase, a user UD's mobile device MD_U needs $13T_h + 4T_{ecm} + T_{eca} + T_{fe} \approx 15.473$ ms, an IoT smart device (sensor node) needs $9T_h + 4T_{ecm} + T_{eca} \approx 11.949$ ms, and a controller node CN requires $12T_h + 6T_{ecm} + 2T_{eca} \approx 4.708$ ms. Table 4.6 compares the computation costs required in different schemes during the "login and authentication phase". Though the proposed scheme needs more computational costs due to utilization of signatures and fuzzy extractor technique, it is justifiable because it provides "superior security and more functionality features as compared to all compared authentication schemes".

Table 4.5: Comparison of security and functionality features

Features	Ali *et al.* [52]	Dhillon and Kalra [122]	Sadhukhan *et al.* [274]	Shuai *et al.* [286]	Proposed
Anonymity	×	✓	×	✓	✓
Untraceability	×	✓	×	✓	✓
User device revocation	✓	×	×	×	✓
Dynamic node addition	✓	✓	×	✓	✓
User biometric change	×	×	×	N/A	✓
User password change	✓	✓	✓	✓	✓
User impersonation attack	×	×	×	×	✓
Stolen smart card/mobile device attack	×	×	×	×	✓
Ephemeral secret leakage (ESL) attack	×	×	×	×	✓
Privileged insider attack	×	×	×	×	✓
Replay attack	✓	✓	✓	✓	✓
Man-in-the-middle attack	✓	✓	✓	✓	✓
Mutual authentication	✓	✓	✓	✓	✓
Unauthorised login detection	✓	✓	✓	✓	✓
DoS attack	×	×	×	✓	✓
Offline guessing attacks	✓	✓	✓	×	✓

Note: N/A: not applicable in a scheme

4.6.3 Comparison of communication costs

For the comparative analysis on communication costs among the proposed scheme and other existing competing schemes, we consider the login and authentication phase. The identities and random secrets are taken to be 160 bits each. The length of output of hash function, the ciphertext block of "symmetric key encryption/decryption using AES-128 [8]" are taken as 256 bits and 128 bits. A point $P = (x_P, y_P)$ on the elliptic curve is taken as (160+160) = 320 bits, with the coordinates x_P and y_P considered as 160 bits each, assuming that "160-bit ECC provides the same security level as that for 1024-bit RSA public key cryptosystem" [66]. Moreover, the timestamp is taken as 32 bits.

In the proposed scheme, the messages $Msg_1 = \langle TID_U, M_1, Y_1, Sign_{y_1}, T_1 \rangle$ with $|TID_U| =$

Table 4.6: Comparison of computational costs

Schemes	User	Sensor node	Controller node
Ali *et al.* [52]	$7T_h + T_{fe} + T_{senc} + T_{sdec}$ ≈ 4.483 ms	$T_{sdec} + 4T_h$ $T_{sdec} + \approx 1.25$ ms	$8T_h + 5T_{enc}/T_{dec}$ ≈ 0.445 ms
Dhillon and Kalra [122]	$10T_h$ ≈ 3.09 ms	$6T_h$ ≈ 1.854 ms	$7T_h$ $\approx 0.0.385$ ms
Sadhukhan *et al.* [274]	$2T_h + T_{senc} + T_{sdec} + T_{ecm}$ ≈ 2.938 ms	$2T_h + T_{senc} + T_{sdec} + T_{ecm}$ ≈ 2.938 ms	$2T_h + 2T_{dec} + 2T_{enc}$ ≈ 0.114 ms
Shuai *et al.* [286]	$8T_h + 2T_{exp}$ ≈ 2.928 ms	$5T_h + T_{exp}$ ≈ 1.773 ms	$7T_h + T_{exp}$ ≈ 0.457 ms
Proposed	$13T_h + 4T_{ecm} + T_{eca} + T_{fe}$ ≈ 15.473 ms	$9T_h + 4T_{ecm} + T_{eca}$ ≈ 11.949 ms	$12T_h + 6T_{ecm} + 2T_{eca}$ ≈ 4.708 ms

Table 4.7: Comparison of communicational overheads

Schemes	Total messages	Total cost (in bits)
Ali *et al.* [52]	5	5504
Dhillon and Kalra [122]	4	4016
Sadhukhan *et al.* [274]	4	5248
Shuai *et al.* [286]	4	7616
Proposed	4	5792

160 bits, $|M_1| = 320$ bits, $|Y_1| = 320$ bits, $|Sign_{y_1}| = 256$ bits and $|T_1| = 32$ bits, requires 1088 bits; $Msg_2 = \langle\ TID_C,\ M_2,\ Y_1,\ Y_2,\ Sign_{y_2},\ T_2\rangle$ with $|TID_C| = 160$ bits, $|M_2| = 256$ bits, $|Y_1| = 320$ bits, $|Y_2| = 320$ bits, $|Sign_{y_2}| = 256$ bits and $|T_2| = 32$ bits, needs 1344 bits; $Msg_3 = \langle\ TID_S^*,\ M_3,\ M_4,\ Y_3,\ Sign_{y_3},\ T_3\ \rangle$ with $|TID_S^*| = 256$ bits, $|M_3| = 256$ bits, $|M_4| = 256$ bits, $|Y_3| = 320$ bits, $|Sign_{y_3}| = 256$ bits and $|T_3| = 32$ bits, demands 1376 bits; and $Msg_4 = \langle TID_U^*, M_3,\ M_4,\ M_5,\ Y_3,\ Y_4,\ Sign_{y_4},\ T_3,\ T_4\rangle$ with $|TID_U^*| = 256$ bits, $|M_3| = 256$ bits, $|M_4| = 256$ bits, $|M_5| = 256$ bits, $|Y_3| = 320$ bits, $|Y_4| = 320$ bits, $|Sign_{y_4}| = 256$ bits, $|T_3| = 32$ bits and $|T_4| = 32$ bits, requires 1984 bits. The cumulative computation cost

in the proposed scheme then amounts to 5792 bits. Table 4.7 compares the communication costs in terms of number of exchanged messages and number of bits required. The existing schemes of Ali *et al.* [52], Dhillon and Kalra [122], Sadhukhan *et al.* [274] and Shuai *et al.* [286] require the communication costs of 5504 bits, 4016 bits, 5248 bits and 7616 bits, respectively. It is noticed that the proposed scheme requires less communication cost as compared to that for the scheme of Shuai *et al.* [286], whereas its communication cost is also comparable with other schemes. This is also acceptable because the proposed scheme provides "superior security and more functionality features as compared to all compared authentication schemes".

4.7 Summary

We designed a novel user authentication scheme in an IoT-enabled intelligent precision agricultural environment. The proposed scheme relies on signatures and ECC. For user biometric verification, the fuzzy extractor technique has been applied that is verified using the Hamming distance in order to avoid false acceptance and false rejection errors. The proposed scheme also supports several functionality features including user mobile device revocation phase, dynamic IoT smart device addition phase and user credentials change phase. The proposed scheme is then shown to be provably secure under the widely-accepted ROR model. The formal security verification using the software automated validation tool, called AVISPA tool, has been applied to show that the proposed scheme is safe against replay and man-in-the-middle attacks. In addition, non-mathematical (informal) security analysis has been also carried out to show that the proposed scheme is robust against other potential attacks. Moreover, the proposed scheme preserves anonymity of users, controller nodes and IoT smart devices, and untraceability properties. The testbed experiments for various cryptographic primitives using MIRACL for both server and Raspberry PI 3 settings are performed. Through the detailed comparative analysis, it is shown that the proposed scheme offers superior security and more functionality features, and has comparable communication and computational overheads as compared to other existing competing user authentication schemes.

Chapter 5

Private Blockchain-envisioned Drones-assisted Authentication Scheme in IoT-enabled Agricultural Environment

Smart agricultural systems inevitably involve communication among untrustworthy entities. The entities in communication may be users on one end trying to access relevant data from the devices placed on their farms. Chapter 4 presented an efficient solution to user authentication with smart devices prior to sensor data access by the user. When the smart farming system requires various untrustworthy smart devices to communicate for data accumulation, an authentication process must be carried out in advance. This and the subsequent chapters seek multiple solutions to the problem of mutual authentication among smart devices used in smart agricultural systems.

In smart farming, several IoT smart devices can be deployed to monitor the agricultural environment. Drones can be further utilized to collect the data sensed by the IoT smart devices, and even sometimes, they can collect information directly from the specific flying zones. The data are then sent to the Ground Station Server (GSS) by the drones. However, insecure communication among the smart devices, drones, and the GSS make the IoT agriculture environment vulnerable to various potential attacks, including replay, impersonation, man-in-the-middle, privileged-insider, and physical smart devices, and drone capture attacks. Apart from these, we need anonymity and untraceability properties that need to be maintained with high priority so that an adversary can not trace the entities sending

the data securely to the *GSS*. For this goal, we design a "new authentication and key management scheme for IoT-enabled intelligent PA, called AKMS-AgriIoT". Furthermore, the blockchain-based solution has been incorporated with AKMS-AgriIoT to achieve decentralization, immutability, and transparency features. The blocks formed with the encrypted transactions and their respective signatures by the *GSS* are mined by the cloud servers in the blockchain center with the help of the widely accepted "Practical Byzantine Fault Tolerance (PBFT)" consensus algorithm to verify and add the blocks.

Mogili and Deepak [243] presented the various applications of drones in precision agricultural systems. Maes and Steppe [224] studied the various ways of applying uncrewed aerial vehicles (UAV) or drones in the farming scenario. Boursianis *et al.* [90] presented a review of using UAVs in conjunction with IoT in smart farming.

Gonzalez-De-Santos *et al.* [150] pointed out that there are two types of uncrewed ground vehicles (UGV): a) one in which existing agricultural vehicles are automated and reused, b) the other in which mobile platforms are specifically designed to fulfill a designated task in the field. The special-designed mobile platforms include wheeled robots and wheel-legged robots. Scivoli [20] performed detailed research on designing four-wheeled steering (4WS) uncrewed ground vehicle for Agriculture 4.0 based on the Ackermann steering mechanism. Vasudevan *et al.* [320] designed a project with the first goal to design and implement an uncrewed aerial vehicle (UAV) and the second goal to design and implement an uncrewed ground vehicle (UGV) and then use the real-time images from both with indices that provide optimal output. Vu *et al.* [325] realized that UAVs are limited in their airspeed and altitude and cannot capture ground features well, whereas UGVs cannot move rapidly and handle obstacles. Their work aims to present the different ways in which UGV-UAV cooperation can be optimized. A multi-phase approach to perform automated navigation and infield operations using both types of uncrewed vehicles in an unstructured agricultural environment has been proposed in Mammarella *et al.* [226].

Blockchain technology allows storing the data only after a thorough check on its integrity using various cryptographic techniques and achieving a consensus on data content and does not allow any tampering of data once stored. This persistence and auditability of stored data give the confidence to use the correct data when needed later and adds transparency, anonymity, and traceability. Usage of blockchain in agriculture can reduce the uncertainty of the output by increasing its predictability and increasing the profit earned while also reducing resource wastage [59, 236].

Smart agriculture system is nation-level system to collect data from various agricultural

ecosystems and use this data appropriately to make decisions about national food security. When applied on such a large scale, the cost of using blockchain to add security to data becomes trivial as the benefits of immutability, transparency, and decentralization features to the data on the blockchain far outweigh the cost of blockchain. A smart agriculture system is also a sensitive application like fintech and medical applications, but with different requirements of security, privacy and trust. Any breach of data from a smart farm system to an unauthorized third party can affect the production and circulation of food and may lead to biowar attacks on food systems. Such situations attack a fundamental resource of food, that is a basic necessity for human survival.

Blockchain enables any untrusted parties to communicate and accomplish business goals seamlessly by acting as a trust enabler among the parties. It provides tamper-proof auditability and traceability of data while ensuring data privacy and anonymity giving control over the logic of time-stamped data sharing. These features are essential in the sensitive IoT network of smart agriculture where the collaboration among untrusted external parties is inevitable. Private blockchain in particular has the ability to validate sensitive data among a closed set of validator nodes in a limited trust environment.

Smart farming systems need to store certain critical data in encrypted format in order to protect their confidentiality. This can be achieved by securely collecting data from the sensors and storing them on a private blockchain. This chapter explores the use of private blockchains and drone technology in alliance with a novel mutual authentication scheme for IoT-based farming.

5.1 System models

5.1.1 Network model

The architecture developed for the IoT-enabled drones-assisted agriculture environment depicted in Figure 5.1 is divided into m flying zones FZ_p ($p = 1, 2, \cdots, m$), with each zone consisting of a number of IoT smart devices SD_j ($j = 1, 2, \cdots, n_{sd}$) and they are monitored by a drone DR_i ($i = 1, 2, \cdots, n_{dr}$), where n_{sd} and n_{dr} denote the number of deployed smart devices and drones, respectively. The Control Room (CR) registers the Ground Station Server (GSS), drones, and smart devices before deployment in the IoT-enabled agriculture environment. During the authentication phase, a drone and its associated smart devices perform mutual authentication prior to data transmission by the smart devices using the

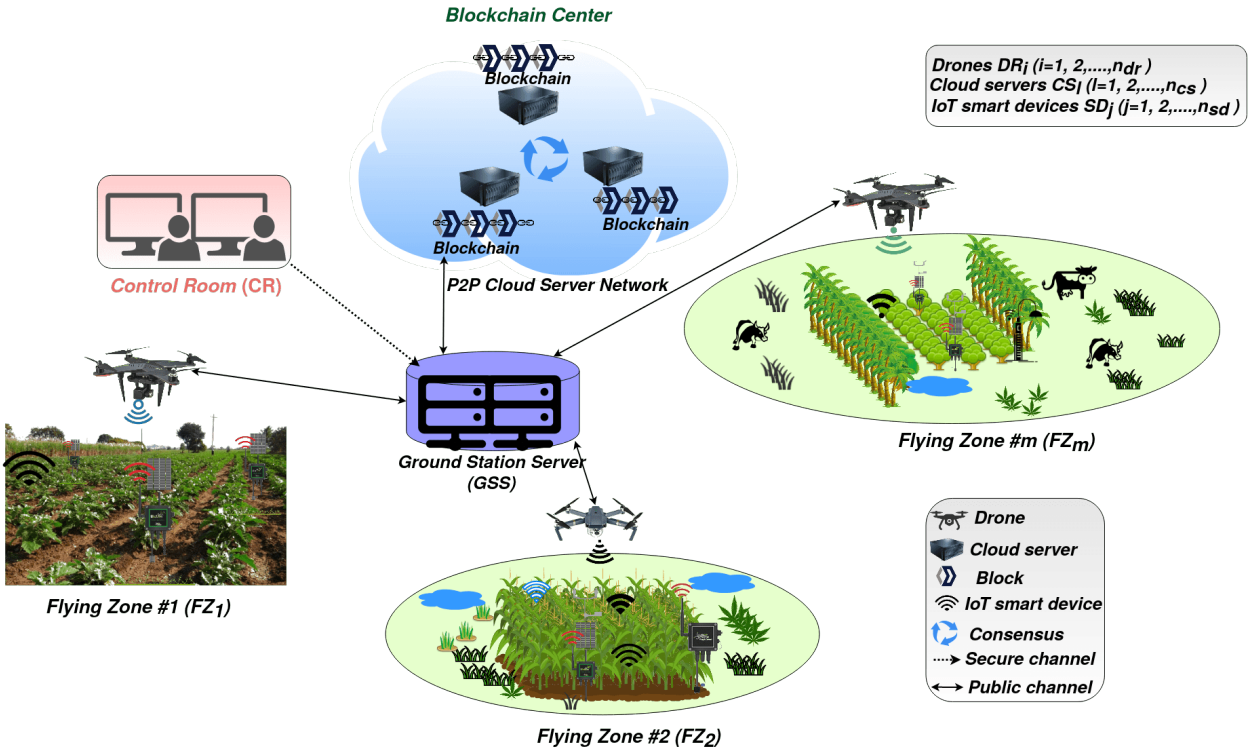

Figure 5.1: Blockchain-envisioned IoT-enabled agricultural environment using drones

established session keys. Similarly, the drones and the GSS are involved in the key management phase for secure communication among them. The blocks formed by the GSS based on the data are securely received from the drones. The encrypted transactions and their signatures created by the GSS are used by the cloud server(s) for forming blocks, and these are added after verification by other cloud servers in a Peer-to-Peer (P2P) cloud servers network using a consensus algorithm. Finally, the blocks are stored by the cloud servers in a decentralized manner in the blockchain center.

5.1.2 Threat model

The threat models "Dolev-Yao (DY) threat model" [125] along with the contemporary *de facto* "Canetti and Krawczyk's model (CK-adversary model)" [94] discussed in Section 1.4.1 of Chapter 1 are applied. The smart devices and drones are untrustworthy entities. The CR is a fully trusted registration authority. The GSS is a semi-trusted entity. A is assumed to have the following capabilities:

- $\mathcal{A}$ can to seize, remove, modify and re-transmit existing messages or circulate coun-

terfeit messages for any communication in public channels between IoT smart devices, drones and GSS.

- Impersonation of IoT smart devices or drones or the GSS may be carried out by $\mathcal{A}$ to perform actions on their behalf.

- Simultaneous multiple executions of the protocol may be initialized by $\mathcal{A}$. The IoT smart devices, drones and GSS may take part in any number of such concurrent executions at the same time.

- The IoT smart devices, drones and GSS are honest and stateless. $\mathcal{A}$ is stateful.

- Hijacking of session states during communication among IoT smart devices, drones and GSS may be employed by $\mathcal{A}$ to extract secret credentials.

In addition, physical capture of smart IoT devices and drones is employed by $\mathcal{A}$ using power analysis attacks [200, 235] and timing attacks [199] to extract secret credentials from their memory. The adversary is not capable of compromising the GSS since they are put under a physical locking system as suggested in [75, 335]. The secret credentials in the GSS will be stored in their secure databases so that the stolen verifier attack is prevented. Thus, the adversary will have no access to the credentials stored in the GSS through the stolen verifier attack so that no other attacks, such as GSS impersonation attack can be launched through the stolen verifier attacks.

5.2 Research contributions

The following are the main contributions to this work:

- We designed a "new authentication and key management scheme for IoT-enabled IPA, called AKMS-AgriIoT". AKMS-AgriIoT is supported by the blockchain solution. The sensing data gathered by the drones in the flying zones from the deployed IoT smart devices are securely transmitted to the GSS. The encrypted transactions and their signatures created by the GSS are used by the cloud server(s) for forming blocks, these are added after verification by "other cloud servers in the P2P CS network" using a consensus algorithm.

- AKMS-AgriIoT is shown to be robust against various potential attacks needed in an IoT-enabled IPA through the formal security analysis and informal (non-mathematical) security analysis.
- Through the formal security verification using the broadly-accepted "Automated Validation of Internet Security Protocols and Applications (AVISPA)" [9], it is shown that AKMS-AgriIoT is also safe against passive/active adversaries.
- We perform experiments of various cryptographic primitives using the widely-accepted "Multiprecision Integer and Rational Arithmetic Cryptographic Library (MIRACL)" [38] to measure the execution time needed for the cryptographic primitives.
- A detailed comparative study on "security and functionality features", "communication costs" and "computational costs" among AKMS-AgriIoT and other relevant existing schemes shows the superiority of AKMS-AgriIoT over existing schemes.
- A blockchain-based implementation of the proposed scheme has been conducted to measure the computational time needed for the varied number of transactions per block and also the varied number of blocks mined in the blockchain.

5.3 The proposed private blockchain-based scheme

In this section, we propose a new authenticated key management scheme for IoT-enabled drones-assisted agricultural environment, called AKMS-AgriIoT, based on the network model provided in Figure 5.1. We apply the current timestamps and random nonces to achieve strong replay attack protection in AKMS-AgriIoT. For this issue, all the communicating entities deployed in the network are assumed to be synchronized with their respective clocks, which is also a common belief used in other networks too [181, 342]. AKMS-AgriIoT involves various phases: a) *system initialization* phase that selects the system parameters for the entities in the network, b) *registration* phase for enrolling the Ground Station Server (GSS), drones and IoT smart devices by the trusted Control Room (CR), c) *authentication* phase for authenticating a drone and its respective IoT smart devices in a flying zone and then generating session (secret) keys among them after mutual authentication, d) *key management* phase helps in establishing secret keys between the GSS and its respective drones, e) *block creation, verification and addition in blockchain center* phase helps in creating, verifying and then adding the blocks using the "Practical Byzantine Fault Tolerance (PBFT)" consensus

algorithm [95] by the GSS and the cloud server(s) in the blockchain center (BC), and f) *dynamic nodes addition* phase that permits to deploy some new IoT smart devices due to the reasons that some IoT smart devices may be physically captured by an adversary or these may be power exhausted. The notations, along with their descriptions listed in Table 5.1 are utilized for analyzing and discussing the proposed AKMS-AgriIoT.

It is worth noticing that the fully trusted control room (CR) is only involved during the registration phase, which is a one-time process. The CR does not also involve in any active participating role during the "authentication phase", "key management phase", and "blockchain construction and addition phase" in our proposed AKMS-AgriIoT. Furthermore, the cloud servers do not have any knowledge of information that the entities exchange during the "authentication phase" and "key management phase", including the established session keys among network entities. As a result, it will be certainly a risky task if we involve the cloud servers for the registration of different network entities, "authentication phase" and "key management phase", and in this situation, several other active attacks, such as "privileged insider attack", "illegal credential leakage attack" and "unauthorized session key computation attack" may be feasible. Thus, we use only the trusted CR for the registration of various network entities instead of the cloud servers in this chapter.

In this work, blockchain technology has been adopted in the proposed system in order to support the authentication and key management designed to provide an effective way to store data (information) in the form of transactions. Blockchain technology provides very strong support after the completion of the execution of the authentication and key management phases as it allows the authenticated credentials to be stored in the blockchain that becomes tamper-proof due to the immutability property of the blockchain. The credentials are verified thoroughly using an appropriate consensus algorithm before being permanently stored in the blockchain. Once stored in the blockchain, the stored credentials cannot be modified in any possible way, thereby increasing the confidence that the details cannot be used by an attacker to launch an impersonation attack and man-in-the-middle attack. In the absence of blockchain, even though our designed schemes are strongly secure against both impersonation and man-in-the-middle attacks, an attacker may break into the storage system used for storing the credentials to obtain valid credentials and use them to extract sensitive data. Our system is strong against single-point failure as the data is stored in a decentralized manner and replicated among multiple cloud servers.

A detailed description of each phase is given below.

Table 5.1: Notations and their description

Notation	Significance
$E_q(u, v)$	A non-singular elliptic curve of the form: $y^2 = x^3 + ux + v \pmod q$ with $4u^3 + 27v^2 \neq 0 \pmod q$
G	A base point in $E_q(u, v)$ of order is n_G as big as q
$x \cdot G$	Elliptic curve point multiplication: $x \cdot G = G + G + \cdots + G\,(x\,times)$
$P + Q$	Elliptic curve point addition; $P, Q \in E_q(u, v)$
CR, ID_{CR}	Control Room (a fully trusted authority) and its identity
GSS	Ground Station Server
TID_{GSS}, PID_{GSS}	GSS's temporary and pseudo identities, respectively
r_{GSS}, R_{GSS}	GSS's random secret and public parameter, respectively
k_{GSS}, Pub_{GSS}	GSS's private and public keys, respectively
DR_i, ID_{DR_i}	i^{th} drone and its real identity
TID_{DR_i}, PID_{DR_i}	DR_i's temporary and pseudo identities, respectively
r_{DR_i}, R_{DR_i}	DR_i's random secret and public parameter, respectively
k_{DR_i}, Pub_{DR_i}	DR_i's private and public keys, respectively
mk_{CR}, Pub_{CR}	CR's private master key and its public key, respectively
$Cert_{GSS}, Cert_{DR_i}$	Certificates of GSS and DR_i created by the CR, respectively
CS_l	l^{th} cloud server in the blockchain center (BC)
k_{CS_l}, Pub_{CS_l}	CS_l's private and public keys, respectively
SD_j, ID_{SD_j}	j^{th} IoT smart device and its real identity
TID_{SD_j}, PID_{SD_j}	SD_j's temporary and pseudo identities, respectively
k_{SD_j}, Pub_{SD_j}	SD_j's private and public keys, respectively
$f(x, y)$	A symmetric bivariate t-degree polynomial over the Galois field $GF(q)$: $f(x, y) = \sum_{i=0}^{t} \sum_{j=0}^{t} a_{ij} x^i y^j$, where $a_{ij} \in Z_q = \{0, 1, 2, \cdots, q-1\}$
RTS_X	Registration timestamp issued by the CR to an entity X
$\|\|$	Concatenation operation
TS_X	Current timestamp generated by an entity X
ΔT	Maximum transmission delay related to a message
$h(\cdot)$	"Collision-resistant cryptographic one-way hash function"
FZ_m	m^{th} flying zone in an agricultural environment

5.3.1 System initialization phase

The fully trusted control room (CR) picks all the related system parameters with the help of the following steps:

- Step SI_1: The CR picks a large prime q and a non-singular elliptic curve of the form: $E_q(u, v) : y^2 = x^3 + ux + v \pmod{q}$ over the Galois field $GF(q)$, with a "point at infinity (zero point)" $\mathcal{O}$, constants $u, v \in Z_q = \{0, 1, 2, \cdots, q-1\}$ such that $4u^3 + 27v^2 \neq 0 \pmod{q}$ is satisfied. Next, the CR chooses a base point $G \in E_q(u, v)$ of order n_G as large as q.

- Step SI_2: The CR picks a "collision-resistant one-way cryptographic hash function", say $h(\cdot)$ (for example SHA-256 hashing algorithm [232]), the "elliptic curve digital signature algorithm (ECDSA)" [186], and the widely-accepted PBFT consensus algorithm [95].

- Step SI_3: The CR also picks a master private key $mk_{CR} \in Z_q^*$ and calculates the corresponding public key $Pub_{CR} = mk_{CR} \cdot G$.

Finally, the CR keeps mk_{CR} as its own private key and makes $\{Pub_{CR}, E_q(u, v), G, h(\cdot), ECDSA.sig, ECDSA.ver, PBFT\}$ as public, where $ECDSA.sig$ and $ECDSA.ver$ are respectively the ECDSA signature and verification algorithms.

5.3.2 Registration phase

This phase is executed by the trusted CR for registering the GSS, the drones, and also the IoT smart devices. It is worth noticing that the registration is a one-time process that is carried by the CR, and it is only involved during this process.

1) **GSS registration:** The following steps are essential to complete the GSS registration process:

   - Step GSR_1: The CR generates a symmetric t-degree bivariate polynomial $f(x, y) = \sum_{i=0}^{t} \sum_{j=0}^{t} a_{ij} x^i y^j$ over the finite field $GF(q)$ where $a_{ij} \in Z_q$, and $f(x, y) = f(y, x)$. The degree t of $f(x, y)$ is chosen much higher than the number of deployed drones in the network in order to provide "unconditional security and t-collusion resistant property against drones capture attack" [80, 112]. Next, the CR computes a polynomial share $f(PID_{GSS}, y)$ using the pseudo-identity $PID_{GSS} = h(ID_{GSS}$

$||mk_{CR}\ ||RTS_{GSS})$, where ID_{GSS} and RTS_{GSS} are the unique identity and registration timestamp of the GSS generated by the CR, respectively.

- Step GSR_2: The CR generates a random secret $r_{GSS} \in Z_q^*$ and its corresponding public $R_{GSS} = r_{GSS} \cdot G$ for the GSS. After that the CR creates a certificate for the GSS as $Cert_{GSS} = r_{GSS} + h(PID_{GSS}\ ||ID_{CR}\ ||R_{GSS}||\ Pub_{CR}) * mk_{CR} \pmod q$. The CR then securely sends the credentials $\{PID_{GSS},\ f(PID_{GSS}, y),\ Cert_{GSS},\ ID_{CR}\}$ to the GSS, and makes R_{GSS} as public. In addition, the CR deletes r_{GSS}.
- Step GSR_3: After receiving the registration information securely from the CR, the GSS generates its own private key $k_{GSS} \in Z_q^*$ and the corresponding public key $Pub_{GSS} = k_{GSS} \cdot G$. Finally, the credentials $\{PID_{GSS},\ f(PID_{GSS}, y),\ Cert_{GSS},\ ID_{CR},\ (k_{GSS},\ Pub_{GSS})\}$ are stored in the GSS's database.

This phase is briefed in Figure 5.2.

Control Room (CR)	Ground Station Server (GSS)
Generate ID_{GSS}, RTS_{GSS}.	
Generate a random secret $r_{GSS} \in Z_q^*$.	
Compute $R_{GSS} = r_{GSS} \cdot G$,	
$PID_{GSS} = h(ID_{GSS}\ ||mk_{CR}\ ||RTS_{GSS})$,	
$Cert_{GSS} = r_{GSS} + h(PID_{GSS}\ ||ID_{CR}$	
$\quad ||R_{GSS}||\ Pub_{CR}) * mk_{CR} \pmod q$.	
Delete r_{GSS}.	
Make R_{GSS} as public.	
$\{PID_{GSS},\ f(PID_{GSS}, y),\ Cert_{GSS},\ ID_{CR}\} \longrightarrow$	
(via secure channel)	
	Generate $k_{GSS} \in Z_q^*$.
	Calculate $Pub_{GSS} = k_{GSS} \cdot G$.
	Store $\{PID_{GSS},\ f(PID_{GSS}, y),$
	$Cert_{GSS},\ ID_{CR},\ (k_{GSS},\ Pub_{GSS})\}$.

Figure 5.2: Summary of GSS registration phase

2) **Drones registration:** To enroll a drone DR_i in a particular flying zone, the CR proceeds with the following steps:

- Step DR_1: The CR first picks a unique real identity ID_{DR_i} for DR_i, calculates its pseudo-identity $PID_{DR_i} = h(ID_{DR_i} \,||mk_{CR}\,||RTS_{DR_i})$ and also selects its random temporary identity TID_{DR_i}, where RTS_{DR_i} is the registration timestamp of DR_i.
- Step DR_2: Next, the CR selects a random secret key $r_{DR_i} \in Z_q^*$ and its respective public $R_{DR_i} = r_{DR_i} \cdot G$. The CR generates a certificate of DR_i as $Cert_{DR_i} = r_{DR_i} + h(PID_{DR_i}\,||ID_{CR}\,||R_{DR_i}||\ Pub_{CR}) * mk_{CR} \pmod q$, and computes a polynomial share $f(PID_{DR_i}, y)$ for the deployed DR_i. In addition, the CR also generates a private key $k_{DR_i} \in Z_q^*$ and the corresponding public key $Pub_{DR_i} = k_{DR_i} \cdot G$.
- Step DR_3: Finally, the CR pre-loads the credentials $\{(TID_{DR_i},\ PID_{DR_i}),\ f(PID_{DR_i}, y),\ Cert_{DR_i},\ ID_{CR},\ (k_{DR_i},\ Pub_{DR_i})\}$. The CR sends securely the credentials $\{TID_{DR_i},\ PID_{DR_i}\}$ to the GSS for each deployed DR_i. Furthermore, the CR also deletes r_{DR_i}.

This phase is briefed in Figure 5.3.

3) **IoT smart devices registration:** Prior to the deployment of IoT smart devices SD_j in the agriculture field, CR registers them with the following steps:

- Step SD_1: The CR first picks a unique real identity ID_{SD_j} for SD_j, calculates its pseudo-identity $PID_{SD_j} = h(ID_{SD_j}\,||mk_{CR}\,||RTS_{SD_j})$ and also selects its random temporary identity TID_{SD_j}, where RTS_{SD_j} is the registration timestamp of SD_j.
- Step SD_2: Next, the CR generates a private key $k_{SD_j} \in Z_q^*$ and the corresponding public key $Pub_{SD_j} = k_{SD_j} \cdot G$. Also CR makes Pub_{SD_j} as public.
- Step DR_3: Finally, the CR pre-loads the credentials $\{(TID_{SD_j},\ PID_{SD_j}),\ (k_{SD_j},\ Pub_{SD_j})\}$. The CR sends securely the credentials $\{TID_{SD_j},\ PID_{SD_j}\}$ to the associated DR_i in a flying zone FZ_i.

This phase is also summarized in Figure 5.4.

5.3.3 Authentication phase

The following steps elaborate the authentication process among the smart device SD_j and drone DR_i:

- Step AP_1: SD_j generates a random number $a_{SD_j} \in Z_q^*$ and a current timestamp TS_{SD_j}, and compute $A_{SD_j} = h(a_{SD_j}\,||TID_{SD_j}\,||PID_{SD_j}\,||k_{SD_j}\,||TS_{SD_j}) \cdot G$, and generates the

Control Room (CR)	Drone (DR_i)
Generate ID_{DR_i}, RTS_{DR_i}. Generate a random secret $r_{DR_i} \in Z_q^*$. Calculate $R_{DR_i} = r_{DR_i} \cdot G$, $PID_{DR_i} = h(ID_{DR_i} \|\|mk_{CR} \|\|RTS_{DR_i})$. Generate TID_{DR_i}. Create certificate as $Cert_{DR_i} = r_{DR_i} + h(PID_{DR_i} \|\|ID_{DR_i}$ $\|\|R_{DR_i}\|\| Pub_{CR}) * mk_{CR} \pmod q$. Generate $k_{DR_i} \in Z_q^*$. Compute $Pub_{DR_i} = k_{DR_i} \cdot G$. Delete r_{DR_i} and make R_{DR_i} as public.	
	Store $\{(TID_{DR_i}, PID_{DR_i})$, $f(PID_{DR_i}, y)$, $Cert_{DR_i}$, ID_{CR}, $(k_{DR_i}, Pub_{DR_i})\}$.

Control Room (CR)	Ground Station Server (GSS)
$\{(TID_{DR_i}, PID_{DR_i})\}$ $\xrightarrow{\text{(via secure channel)}}$	

Figure 5.3: Summary of drone registration phase

signature on a_{SD_j} as $Sign_{SD_j} = h(a_{SD_j} \|\|TID_{SD_j} \|\|PID_{SD_j} \|\|k_{SD_j} \|\|TS_{SD_j}) + h(Pub_{SD_j} \|\|Pub_{DR_i} \|\|Pub_{GSS} \|\|TS_{SD_j}) * k_{SD_j} \pmod q$. Next, SD_j sends the message $Msg_{SD_1} = \{TID_{SD_j}, A_{SD_j}, Sign_{SD_j}, TS_{SD_j}\}$ to the DR_i via public channel.

- Step AP_2: After receiving the message Msg_{SD_1} at a time $TS^*_{SD_j}$, DR_i verifies the timestamp as $|TS^*_{SD_j} - TS_{SD_j}| < \Delta T$. If it valid, DR_i verifies the signature as $Sign_{SD_j} \cdot G = A_{SD_j} + h(Pub_{SD_j} \|\|Pub_{DR_i} \|\|Pub_{GSS} \|\|TS_{SD_j}) \cdot Pub_{SD_j}$. If it is valid, DR_i generates a current timestamp $TS_{DR_{i_1}}$ and a random number $b_{DR_i} \in Z_q^*$, and compute $B_{DR_i} = h(b_{DR_i} \|\|TID_{DR_i} \|\|PID_{DR_i} \|\|k_{DR_i} \|\|TS_{DR_{i_1}}) \cdot G$. Next, DR_i compute the elliptic curve Diffie-Hellman type key as $DHK_{DR_i,SD_j} = h(b_{DR_i} \|\|TID_{DR_i} \|\|PID_{DR_i} \|\|k_{DR_i} \|\|TS_{DR_{i_1}}) \cdot A_{SD_j}$, and the session key $SK_{DR_i,SD_j} = h(DHK_{DR_i,SD_j} \|\|Sign_{SD_j} \|\|TS_{SD_j} \|\|TS_{DR_{i_1}})$. After that, DR_i computes the signature on b_{DR_i} and SK_{DR_i,SD_j} as $Sign_{DR_{i_1}} = h(b_{DR_i} \|\|TID_{DR_i} \|\|PID_{DR_i} \|\|k_{DR_i} \|\|TS_{DR_{i_1}}) + h(SK_{DR_i,SD_j} \|\|Pub_{SD_j} \|\|Pub_{GSS} \|\|TS_{DR_{i_1}}) * k_{DR_i} \pmod q$. DR_i generates a new temporary identity $TID^{new}_{SD_j}$ for SD_j and calculate

Control Room (CR)	Smart Device (SD_j)
Generate ID_{SD_j}, RTS_{SD_j}.	
Calculate $PID_{SD_j} = h(ID_{SD_j} \|\|mk_{CR} \|\|RTS_{SD_j})$.	
Generate TID_{SD_j}.	
Generate $k_{SD_j} \in Z_q^*$.	
Compute $Pub_{SD_j} = k_{SD_j} \cdot G$.	
Make Pub_{SD_j} as public.	
	Store $\{(TID_{SD_j}, PID_{SD_j}),$ $(k_{SD_j}, Pub_{SD_j})\}$.

Control Room (CR)	Drone (DR_i)
$\{(TID_{SD_j}, PID_{SD_j})\}$ $\xrightarrow{\text{(via secure channel)}}$	

Figure 5.4: Summary of smart device registration phase

$TID^*_{SD_j} = TID^{new}_{SD_j} \oplus h(TID_{SD_j} \|\|SK_{DR_i,SD_j} \|\|Sign_{DR_{i_1}} \|\|TS_{DR_{i_1}})$. DR_i sends the message $Msg_{SD_2} = \{TID^*_{SD_j}, B_{DR_i}, Sign_{DR_{i_1}}, TS_{DR_{i_1}}\}$ to the SD_j via public channel.

- Step AP_3: After receiving the message Msg_{SD_2} at a time $TS^*_{DR_{i_1}}$, SD_j validates the timestamp as $|TS^*_{DR_{i_1}} - TS_{DR_{i_1}}| < \Delta T$. If it valid, SD_j computes the elliptic curve Diffie-Hellman type key as $DHK_{SD_j,DR_i} = h(a_{SD_j} \|\|TID_{SD_j} \|\|PID_{SD_j} \|\|k_{SD_j} \|\|TS_{SD_j}) \cdot B_{DR_i}$, and the session key $SK_{SD_j,DR_i} = h(DHK_{SD_j,DR_i} \|\|Sign_{SD_j} \|\|TS_{SD_j} \|\|TS_{DR_{i_1}})$. After that verify the signature $Sign_{DR_{i_1}}$ as $Sign_{DR_{i_1}} \cdot G = B_{DR_i} + h(SK_{SD_j,DR_i} \|\|Pub_{SD_j} \|\|Pub_{GSS} \|\|TS_{DR_{i_1}}) \cdot Pub_{DR_i}$. If it is verified successfully, SD_j retrieves $TID^{new}_{SD_j}$ as $TID^{new}_{SD_j} = TID^*_{SD_j} \oplus h(TID_{SD_j} \|\|SK_{SD_j,DR_i} \|\|Sign_{DR_{i_1}} \|\|TS_{DR_{i_1}})$. Finally SD_j updates TID_{SD_j} by $TID^{new}_{SD_j}$ to its database.

At the end of this phase, both SD_j and DR_i share the same secret session key SK_{SD_j,DR_i} $(= SK_{DR_i,SD_j})$.

5.3.4 Key management phase

This phase involves the key establishment between the GSS and its associated drone (DR_i) for each flying zone FZ_l ($l = 1, 2, \cdots, m$). The following steps need to executed to complete this task:

- Step KM_1: The GSS first generates a random number $x_{GSS} \in Z_q^*$ and current timestamp TS_{GSS}, and calculates $X_{GSS} = h(x_{GSS} \,||k_{GSS} \,||PID_{GSS} \,||TS_{GSS}) \cdot G$, $PID^*_{GSS} = PID_{GSS} \oplus h(PID_{DR_i} \,||ID_{CR} \,||TS_{GSS})$ and a signature on x_{GSS} as $Sign_{x_{GSS}} = h(x_{GSS} \,||k_{GSS} \,||PID_{GSS} \,||TS_{GSS}) + h(Pub_{GSS} \,||Cert_{GSS} \,||PID^*_{GSS} \,||TID_{DR_i}) * k_{GSS} \pmod q$. Next, the GSS sends the message $Msg_{GD1} = \{TID_{DR_i}, X_{GSS}, Cert_{GSS}, Sign_{x_{GSS}}, PID^*_{GSS}, TS_{GSS}\}$ to drone DR_i via open channel.

- Step KM_2: If the message Msg_{GD1} is received at time TS^*_{GSS}, DR_i checks validity of timestamp by $|TS^*_{GSS} - TS_{GSS}| < \Delta T$. If it is valid, DR_i retrieves PID_{GSS} as $PID_{GSS} = PID^*_{GSS} \oplus h(PID_{DR_i} \,||ID_{CR} \,||TS_{GSS})$, and then checks the validity of received TID_{DR_i}, certificate and signature using all the public information by verifying if $Cert_{GSS} \cdot G = R_{GSS} + h(PID_{GSS} \,||ID_{CR} \,||R_{GSS}|| \,Pub_{CR}) \cdot Pub_{CR}$ and $Sign_{x_{GSS}} \cdot G = X_{GSS} + h(Pub_{GSS} \,||Cert_{GSS} \,||PID^*_{GSS} \,||TID_{DR_i}) \cdot Pub_{GSS}$. If the certificate and signature verification pass successfully, the next step is executed; otherwise, the phase is instantly terminated by DR_i.

- Step KM_3: DR_i generates a random number $y_{DR_i} \in Z_q^*$ and current timestamp TS_{DR_i}, and calculates $Y_{DR_i} = h(y_{DR_i} \,||k_{DR_i} \,||PID_{DR_i} \,||TS_{DR_i}) \cdot G$, $f(PID_{DR_i}, PID_{GSS})$ using the retrieved PID_{GSS} corresponding to the GSS, the Diffie-Hellman type secret key $DHK_{DR_i,GSS} = h(y_{DR_i} \,||k_{DR_i} \,||PID_{DR_i} \,||TS_{DR_i}) \cdot X_{GSS}$, the secret shared key with GSS as $SK_{DR_i,GSS} = h(DHK_{DR_i,GSS} \,||f(PID_{DR_i}, PID_{GSS}) \,||Cert_{DR_i} \,||Cert_{GSS})$ and the signature on y_{DR_i} and $SK_{DR_i,GSS}$ as $Sign_{DR_i} = h(y_{DR_i} \,||k_{DR_i} \,||PID_{DR_i} \,||TS_{DR_i}) + h(Pub_{DR_i} \,||Cert_{DR_i} \,||ID_{CR}|| \,SK_{DR_i,GSS}) * k_{DR_i} \pmod q$. DR_i also generates its own new temporary identity $TID^{new}_{DR_i}$ and calculates $TID^*_{DR_i} = TID^{new}_{DR_i} \oplus h(TID_{DR_i} \,||f(PID_{DR_i}, \,PID_{GSS}) \,||SK_{DR_i,GSS} \,||TS_{DR_i})$, and sends the message $Msg_{GD2} = \{TID^*_{DR_i}, Cert_{DR_i}, Y_{DR_i}, Sign_{DR_i}, TS_{DR_i}\}$ to the GSS via public channel. Furthermore, DR_i updates TID_{DR_i} with the newly generated $TID^{new}_{DR_i}$ in its database corresponding to PID_{DR_i}.

- Step KM_4: After receiving Msg_{GD2} at time $TS^*_{DR_i}$, the GSS validates if $|TS^*_{DR_i} - TS_{DR_i}| < \Delta T$. If it is so, validate the certificate $Cert_{DR_i}$ by $Cert_{DR_i} \cdot G = R_{DR_i} + h(PID_{DR_i} \,||ID_{CR} \,||R_{DR_i}|| \,Pub_{CR}) \cdot Pub_{CR}$. If the certificate validation passes, GSS calculates $f(PID_{GSS}, \,PID_{DR_i})$, the Diffie-Hellman type secret key $DHK_{GSS,DR_i} = h(x_{GSS} \,||k_{GSS} \,||PID_{GSS} \,||TS_{GSS}) \cdot Y_{DR_i}$ and the secret shared key with DR_i as $SK_{GSS,DR_i} = h(DHK_{GSS,DR_i} \,||f(PID_{GSS}, \,PID_{DR_i}) \,||Cert_{DR_i} \,||Cert_{GSS})$. If the signature validation passes through the condition: $Sign_{DR_i} \cdot G = Y_{DR_i} + h(Pub_{DR_i}$

$||Cert_{DR_i}\ ||ID_{CR}||\ SK_{GSS,DR_i}) \cdot Pub_{DR_i}$, the GSS computes $TID^{new}_{DR_i} = TID^{*}_{DR_i} \oplus h(TID_{DR_i}\ ||f(PID_{GSS}, PID_{DR_i})\ ||SK_{GSS,DR_i}\ ||TS_{DR_i})$ and updates TID_{DR_i} with the new $TID^{new}_{DR_i}$ in its database corresponding to PID_{DR_i}.

At the end of this phase, both DR_i and GSS share the same secret key $SK_{DR_i,GSS}$ $(= SK_{GSS,DR_i})$. Figure 5.5 illustrates briefly both the phases described in Sections 5.3.3 and 5.3.4.

5.3.5 Block creation, verification and addition in blockchain

In this section, we provide a comprehensive explanation of making the transactions by the GSS and blocks by a cloud server CS in the P2P CS network, namely blockchain center BC. The blockchain consists of a set of blocks connected as a chain and is stored on the cloud servers such that all the cloud servers hold the same copy of the blockchain at any given time by using a voting-based consensus algorithm. We assume that the data collected by the drones from various IoT smart devices in the flying zones are private and confidential. As a result, we want to put the collected data by the GSS from the drones in a private blockchain that is maintained by the P2P CS network. The drones and the smart sensor devices have very limited computational resources. Therefore, delegating the task of creating transactions for the blockchain may turn out to be very taxing on these entities. Due to this purpose, it is deemed more computationally efficient to allow the GSS to generate the transactions to be added to the blockchain. For the block addition into the existing private blockchain by the CS, the block verification is executed through the consensus algorithm. In this chapter, we use the "Practical Byzantine Fault Tolerance (PBFT)" consensus algorithm [95].

The detailed description for doing this task is given below.

- The GSS first securely gathers the agriculture-related information from the drones using their established secret keys in Section 5.3.4. After that the GSS makes several transactions, say n_t transactions Tx_1, Tx_2, $\cdots$, Tx_{n_t} for a block $Block_m$, and then encrypts all these n_t transactions using its own public key Pub_{GSS} as $\{E_{Pub_{GSS}}(Tx_1), E_{Pub_{GSS}}(Tx_2), \cdots, E_{Pub_{GSS}}(Tx_{n_t})\}$.

- Next, the GSS creates a digital signature using the "Elliptic Curve Digital Signature Algorithm (ECDSA)" on all n_t transactions together using its own private key k_{GSS} as $ECDSA.sig_{Block_m} = ECDSA.sig_{k_{GSS}}(M)$, where $M = h(E_{Pub_{GSS}}(Tx_1)||\ E_{Pub_{GSS}}(Tx_2)||\ \cdots ||\ E_{Pub_{GSS}}(Tx_{n_t}))$, where $E(\cdot)$ and $D(\cdot)$ represent ECC encryption

Authentication between a smart device and its associated drone	
Smart Device (SD_j)	Drone (DR_i)
Generate random number $a_{SD_j} \in Z_q^*$, current timestamp TS_{SD_j}. Compute A_{SD_j} and generate signature $Sign_{SD_j}$. $Msg_{SD_1} = \{TID_{SD_j}, A_{SD_j}, Sign_{SD_j}, TS_{SD_j}\}$ $\xrightarrow{\text{(via open channel)}}$	Verify timestamp TS_{SD_j}. If valid, verify signature by $Sign_{SD_j} \cdot G = A_{SD_j} + h(Pub_{SD_j} \|\|Pub_{DR_i} \|\|Pub_{GSS}$ $\|\|TS_{SD_j}) \cdot Pub_{SD_j}$. Generate current timestamp $TS_{DR_{i_1}}$ and random number $b_{DR_i} \in Z_q^*$. Compute B_{DR_i}, $DHK_{DR_i,SD_j} = h(b_{DR_i} \|\|TID_{DR_i} \|\|PID_{DR_i}$ $\|\|k_{DR_i} \|\|TS_{DR_{i_1}}) \cdot A_{SD_j}$, $SK_{DR_i,SD_j} = h(DHK_{DR_i,SD_j} \|\|Sign_{SD_j} \|\|TS_{SD_j} \|\|TS_{DR_{i_1}})$. Compute signature on b_{DR_i} and SK_{DR_i,SD_j} as $Sign_{DR_{i_1}}$ Generate new temporary identity $TID_{SD_j}^{new}$ for SD_j. Calculate $TID_{SD_j}^* = TID_{SD_j}^{new} \oplus h(TID_{SD_j} \|\|SK_{DR_i,SD_j}$ $\|\|Sign_{DR_{i_1}} \|\|TS_{DR_{i_1}})$. $Msg_{SD_2} = \{TID_{SD_j}^*, B_{DR_i}, Sign_{DR_{i_1}}, TS_{DR_{i_1}}\}$ $\xleftarrow{\text{(via open channel)}}$
Verify timestamp $TS_{DR_{i_1}}$. If valid, compute DHK_{SD_j,DR_i} $= h(a_{SD_j} \|\|TID_{SD_j} \|\|PID_{SD_j} \|\|k_{SD_j} \|\|TS_{SD_j}) \cdot B_{DR_i}$, $SK_{SD_j,DR_i} = h(DHK_{SD_j,DR_i} \|\|Sign_{SD_j} \|\|TS_{SD_j} \|\|TS_{DR_{i_1}})$. Verify signature $Sign_{DR_{i_1}}$. If valid, retrieve $TID_{SD_j}^{new} =$ $TID_{SD_j}^* \oplus h(TID_{SD_j} \|\|SK_{SD_j,DR_i} \|\|Sign_{DR_{i_1}} \|\|TS_{DR_{i_1}})$. Update TID_{SD_j} by $TID_{SD_j}^{new}$.	Update TID_{SD_j} by $TID_{SD_j}^{new}$.
Both SD_j and DR_i share the same secret key SK_{SD_j,DR_i} $(= SK_{DR_i,SD_j})$	
Key management between the GSS and its associated drone	
GSS	Drone (DR_i)
Generate random number $x_{GSS} \in Z_q^*$ and current timestamp TS_{GSS}. Calculate X_{GSS}, $PID_{GSS}^* = PID_{GSS} \oplus h(PID_{DR_i} \|\|ID_{CR} \|\|TS_{GSS})$, signature on x_{GSS} as $Sign_{x_{GSS}}$. $Msg_{GD1} = \{TID_{DR_i}, X_{GSS}, Cert_{GSS},$ $Sign_{x_{GSS}}, PID_{GSS}^*, TS_{GSS}\}$ $\xrightarrow{\text{(via open channel)}}$	Check validity of timestamp TS_{GSS}. If so, compute $PID_{GSS} = PID_{GSS}^* \oplus h(PID_{DR_i} \|\|ID_{CR} \|\|TS_{GSS})$. Check validity of received TID_{DR_i}, certificate $Cert_{GSS}$ and signature $Sign_{x_{GSS}}$. If all are valid, generate random number $y_{DR_i} \in Z_q^*$ and current timestamp TS_{DR_i}. Compute Y_{DR_i}, $DHK_{DR_i,GSS} = h(y_{DR_i} \|\|k_{DR_i} \|\|PID_{DR_i}$ $\|\|TS_{DR_i}) \cdot X_{GSS}$, $SK_{DR_i,GSS} = h(DHK_{DR_i,GSS} \|\|f(PID_{DR_i}, PID_{GSS})$ $\|\|Cert_{DR_i} \|\|Cert_{GSS})$, signature on y_{DR_i} and $SK_{DR_i,GSS}$ as $Sign_{DR_i}$. Create new temporary identity $TID_{DR_i}^{new}$. Calculate $TID_{DR_i}^* = TID_{DR_i}^{new} \oplus h(TID_{DR_i}$ $\|\|f(PID_{DR_i}, PID_{GSS}) \|\|SK_{DR_i,GSS} \|\|TS_{DR_i})$. $Msg_{GD2} = \{TID_{DR_i}^*, Cert_{DR_i}, Y_{DR_i}, Sign_{DR_i}, TS_{DR_i}\}$ $\xleftarrow{\text{(via open channel)}}$
Verify timestamp TS_{DR_i}. If valid, validate certificate $Cert_{DR_i}$. If valid, compute $f(PID_{GSS}, PID_{DR_i})$, $DHK_{GSS,DR_i} = h(x_{GSS} \|\|k_{GSS} \|\|PID_{GSS} \|\|TS_{GSS}) \cdot Y_{DR_i}$, $SK_{GSS,DR_i} = h(DHK_{GSS,DR_i} \|\|f(PID_{GSS}, PID_{DR_i})$ $\|\|Cert_{DR_i} \|\|Cert_{GSS})$. Verify signature $Sign_{DR_i}$. If valid, compute $TID_{DR_i}^{new}$. Update TID_{DR_i} with new $TID_{DR_i}^{new}$.	Update TID_{DR_i} with $TID_{DR_i}^{new}$.
Both GSS and DR_i share the same secret key SK_{GSS,DR_i} $(= SK_{DR_i,GSS})$	

Figure 5.5: Summary of authentication and key management phases

Block Header	
Block Version	$BVer_m$
Previous Block Hash	PBH_m
Merkle Tree Root	MTR_m
Timestamp	TS_m
Owner of Block	OB_m (Identity of the CS)
Public Key of Signer GSS	Pub_{GSS}
Block Payload (Encrypted Transactions)	
List of n_t Encrypted Transactions $\#i$ (Tx_i)	$\{E_{Pub_{GSS}}(Tx_i)\|i = 1, 2, \cdots, n_t\}$
Signature on Transactions	$ECDSA.sig_{Block_m}$
Current Block Hash	$CBHash_m$

Figure 5.6: Structure of a block $Block_m$ based on transactions

and decryption, and $ECDSA.sig(\cdot)$ and $ECDSA.ver(\cdot)$ denote the "ECDSA signature generation" and "ECDSA signature verification" algorithms, respectively.

- The GSS sends these n_t encrypted transactions along with their signature as $Msg_{GC} = \{(E_{Pub_{GSS}}(Tx_i)|i = 1, 2, \cdots, n_t), ECDSA.sig_{Block_m}\}$ to a cloud server CS in the blockchain center (BC). After receiving the message Msg_{GC} by the CS from the GSS, the CS will create the block $Block_m$ which contains these encrypted transactions, signature and other necessary objects, such as Merkle tree root on n_t encrypted transactions, previous block hash, public key of GSS and current block hash, that are described in Figure 5.6.

- Once a block $Block_m$ is formed by the CS, the following two tasks will be executed: a) a leader selection by a leader selection algorithm and b) a consensus for block validation as well as addition into the blockchain. It is assumed that a leader is picked in the P2P CS network successfully using the existing mechanism as suggested in [348]. The consensus algorithm (PBFT) will then be executed, and its detailed description is given below. Note that each cloud server CS_l in the BC has a private-public key pair (k_{CS_l}, Pub_{CS_l}), where $k_{CS_l} \in Z_q^*$ is the randomly chosen private key and $Pub_{CS_l} = k_{CS_l} \cdot G$ is its corresponding public key. The public keys of all the cloud servers are known to each other in the BC.

Algorithm 1 Consensus for block validation and addition

1: Assume a leader (L) is selected which has a block of the form: $Block_m = \{BVer_m,\ PBH_m,\ MTR_m,\ TS_m,\ OB_m,\ Pub_{GSS},\ \{E_{Pub_{GSS}}(Tx_i)\ |i = 1,\ 2,\ \cdots, n_t\},\ ECDSA.sig_{Block_m},\ CBHash_m\}$.

2: L generates a current timestamp TS_{CS_l} for each follower cloud server peer node, say CS_l and starts voting process.

3: L computes the encrypted voting request $VTReq$ using CS_l 's public key Pub_{CS_l} as $EVTReq = E_{Pub_{CS_l}}(VTReq,\ TS_{CS_l})$ and the signature on $VTReq$ as $sig_{VTReq} = ECDSA.sig_{k_L}(EVTReq)$ for each follower CS_l, $l = 1,\ 2,\ \cdots,\ n_{cs}$ with $L \neq CS_l$.

4: L then sends the messages containing the same block $Block_m$ with encrypted voting request and signature as $\{Block_m,\ EVTReq,\ Sig_{VTReq}\}$ to each follower CS_l, $(l = 1, 2, \cdots, n_{cs}, L \neq CS_l)$.

5: Assume that message is received from the L by every follower CS_l at time $TS^*_{CS_l}$.

6: **for** each follower node CS_l **do**

7: Verify signature sig_{VTRreq} using $ECDSA.ver$ algorithm.

8: **if** signature is valid **then**

9: Compute $(VTReq, TS_{CS_l}) = D_{k_{CS_l}}[EVTReq]$.

10: **if** $(|TS^*_{CS_l} - TS_{CS_l}| < \Delta T)$ **then**

11: Compute the Merkle tree root, say MTR'_m on the encrypted transactions present in $Block_m$.

12: **if** $(MTR'_m = MTR_m)$ **then**

13: Compute block hash $CBHash'_m$ on all the fields $\{BVer_m,\ PBH_m,\ MTR_m,\ TS_m,\ OB_m,\ Pub_{GSS},\ \{E_{Pub_{GSS}}(Tx_i)\ |i = 1,\ 2,\ \cdots, n_t\},\ ECDSA.sig_{Block_m}\}$.

14: **if** $(CBHash'_m = CBHash_m)$ **then**

15: Send the block validation status $BVStatus$ and vote reply $VTRep$ as $\{E_{Pub_L}(VTRep, BVStatus), sig_{VTRep}\}$ to the leader L, where $sig_{VTRep} = ECDSA.sig_{k_{CS_l}}[E_{Pub_L}(VTRep, BVStatus)]$.

Algorithms 1 and 2 are based on the voting-based PBFT consensus algorithm. It takes the following inputs: a) the block $Block_m$ created by a cloud server CS consisting of the n_t transactions generated by the GSS; b) the private-public key pair $(k_{CS_l},\ Pub_{CS_l})$ of all cloud servers CS_l in the P2P network; and c) the number of faulty nodes $f_{n_{CS}}$ in the Blockchain center. The leader cloud server, say L, first generates multiple encrypted voting requests $EVTReq$ using public keys of the receiver cloud servers CS_l, signs the requests

Algorithm 2 Consensus for block validation and addition (Continued...)

Let $VVCount$ denote the valid vote counter and set $VVCount \leftarrow 0$.
for each received message $\{E_{Pub_L}(VTRep,\ BVStatus),\ sig_{VTRep}\}$ from a responded follower CS_l **do**
  Verify signature sig_{VTRep} using $ECDSA.ver$ algorithm.
  if signature is valid **then**
    Compute $(VTRep, BVStatus) = D_{k_L}\ [E_{Pub_L}(VTRep,\ BVStatus)]$.
    if $((VTRep = valid)$ and $(BVStatus = valid))$ **then**
      Set $VVCount = VVCount + 1$.
if $(VVCount > 2 * f_{n_{cs}} + 1)$ **then**
  Send the commit response to all followers.
  Add block $Block_m$ to the blockchain.

using the $ECDSA$ signature generation algorithm, and sends the requests to the respective follower cloud servers along with the $Block_m$. Every follower cloud server CS_l then verifies the signature on the request using the $ECDSA$ signature verification algorithm, decrypts $EVTReq$ using its own private key k_{CS_l} and verifies the timestamp in the request, the Merkle tree root of the block and the current block hash $CBHash_m$. If all are valid, CS_l sends the status of the above verifications along with its voting reply encrypted with the public key of the leader L and $ECDSA$-based signature on the reply. The leader L verifies the signature and counts the votes (maintained with the counter $VVCount$) only if both the block verification and the voting reply are valid. Once all replies are received and if $VVCount > 2 * f_{n_{cs}} + 1$, L sends a commit block command to the follower nodes by L. Finally, the block $Block_m$ is added to the respective distributed ledgers of the peer nodes.

Finally, the overview of the proposed AKMS-AgriIoT has been provided in Figure 5.7.

5.3.6 Dynamic nodes addition phase

This phase allows the addition of new IoT smart devices due to their power exhaustion or physical device capturing issues by an adversary. Therefore, to add a new smart device, say SD_j^{new} in a specific flying zone, say FZ_i, we need the following steps:

- Step NSD_1: The CR selects a unique real identity $ID_{SD_j}^{new}$ for SD_j^{new}, computes its pseudo-identity $PID_{SD_j}^{new} = h(ID_{SD_j}^{new} \,||mk_{CR}\,||RTS_{SD_j}^{new})$ and also picks its random temporary identity $TID_{SD_j}^{new_1}$, where $RTS_{SD_j}^{new}$ is the registration timestamp of SD_j^{new}.

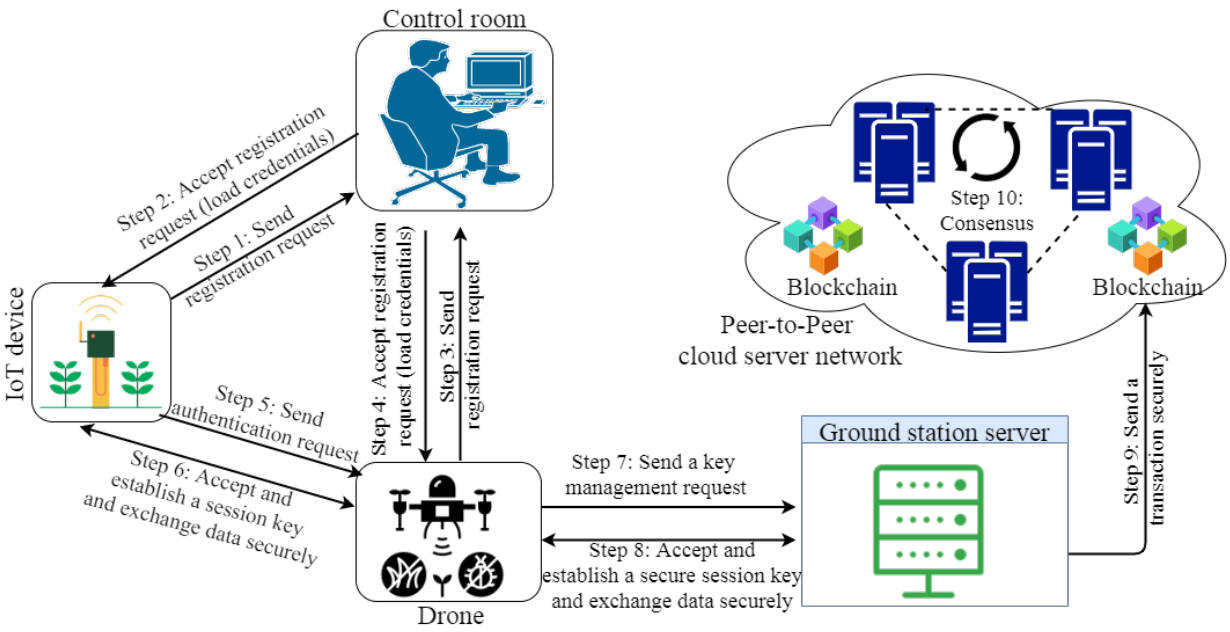

Figure 5.7: Overall process diagram of the proposed scheme

- Step NSD_2: The CR then generates a private key $k_{SD_j}^{new} \in Z_q^*$ and calculates the corresponding public key $Pub_{SD_j}^{new} = k_{SD_j}^{new} \cdot G$. In addition, the CR also makes $Pub_{SD_j}^{new}$ as public.

- Step NSD_3: Finally, the CR pre-loads the credentials $\{(TID_{SD_j}^{new_1}, PID_{SD_j}^{new}), (k_{SD_j}^{new}, Pub_{SD_j}^{new})\}$. The CR needs to send securely the credentials $\{TID_{SD_j}^{new_1}, PID_{SD_j}^{new}\}$ to the associated DR_i of the deployed SD_j^{new} in the respective flying zone.

This phase is also summarized in Figure 5.8.

5.4 Security analysis

In this section, we perform a detailed formal as well as informal (non-mathematical) security analysis to exhibit that the proposed scheme (AKMS-AgriIoT) has the ability to resist the following potential attacks that are essential in an IoT environment:

Wang *et al.* [327] made an important observation that the broadly-accepted formal mechanisms for security analysis, such as "random oracle model" and "Burrows-Abadi-Needham (BAN) logic [92]" can not capture the "structural mistakes" in a designed user authentication scheme. As a result, guaranteeing the soundness of the designed authentication protocols

Control Room (CR)	Smart Device (SD_j^{new})
Generate $ID_{SD_j}^{new}$, $RTS_{SD_j}^{new}$.	
Calculate $PID_{SD_j}^{new} = h(ID_{SD_j}^{new}$	
$\|\|mk_{CR}\|\|RTS_{SD_j}^{new})$.	
Generate $TID_{SD_j}^{new_1}$.	
Generate $k_{SD_j}^{new} \in Z_q^*$.	
Calculate $Pub_{SD_j}^{new} = k_{SD_j}^{new} \cdot G$.	
Make $Pub_{SD_j}^{new}$ as public.	
	Store $\{(TID_{SD_j}^{new_1}, PID_{SD_j}^{new}), (k_{SD_j}^{new}, Pub_{SD_j}^{new})\}$.

Control Room (CR)	Drone (DR_i)
$\{TID_{SD_j}^{new_1}, PID_{SD_j}^{new}\}$ $\xrightarrow{\text{(via secure channel)}}$	

Figure 5.8: Summary of dynamic smart device addition phase

still is an open issue. Due to this observation, we require other kinds of security analysis, including informal security analysis and formal security verification using automated software verification tools so that the designed authentication protocols will be secure against various potential attacks with very high probability.

5.4.1 Formal security analysis under ROR model

The widely-accepted "Real-Or-Random (ROR)" oracle model [43] has been applied to show the session key (secret key) security between a smart device SD_j and a drone DR_i in the authentication phase, and also between the GSS and DR_i in the key management phase illustrated in Figure 5.5 for the proposed AKMS-AgriIoT against an adversary $\mathcal{A}$.

In the following, we first briefly provide the description of the ROR model. $\mathcal{A}$ will have access to various queries that are illustrated in Table 5.2. In addition, as in [97], a "collision-resistant one-way cryptographic hash function $h(\cdot)$" is modeled as a random oracle, say $Hash$ that is provided to all the engaged members including $\mathcal{A}$.

Participants. We have three participants, namely DR_i, SD_j and GSS during the authentication and key management phases of our proposed AKMS-AgriIoT. The entities SD_j and DR_i take participation during the authentication phase to establish a session key

Table 5.2: Queries and their functions

Query	Description
$Execute(\Pi_{DR_i}^{l_1}, \Pi_{GSS}^{l_2}, \Pi_{SD_j}^{l_3})$	This query is executed by $\mathcal{A}$ to intercept the messages communicated between DR_i, SD_j and GSS
$CorruptDR(\Pi_{DR_i}^{l_1})$	$\mathcal{A}$ executes such a query to extract "secret credentials stored in a compromised drone DR_i"
$CorruptSD(\Pi_{SD_j}^{l_3})$	$\mathcal{A}$ executes such a query to extract "secret credentials stored in a compromised smart device SD_j"
$Reveal(\Pi^l)$	This query is executed by $\mathcal{A}$ to disclose the session key SK_{SD_j,DR_i} $(= SK_{DR_i,SD_j})$ and secret key $SK_{DR_i,GSS}$ $(= SK_{GSS,DR_i})$ between Π^l and its respective partner
$Test(\Pi^l)$	$\mathcal{A}$ executes this query to validate the revealed session key SK_{SD_j,DR_i} $(= SK_{DR_i,SD_j})$ between SD_j and DR_i, and secret key $SK_{DR_i,GSS}$ $(= SK_{GSS,DR_i})$ between DR_i and GSS by utilizing a "random outcome of a flipped unbiased coin, c"

(see Section 5.3.3), whereas the entities GSS and its associated drone (DR_i) are involved during the key management phase (see Section 5.3.4). Let $\Pi_{DR_i}^{l_1}$, $\Pi_{GSS}^{l_2}$ and $\Pi_{SD_j}^{l_3}$) denote the l_1^{th}, l_2^{nd} and l_3^{rd} instances of DR_i, GSS and SD_j, respectively, which are termed as the "random oracles".

Accepted state. An instance Π^l is said to be in an "accepted state" if it receives the last valid communicated message. If all the communicated messages in a particular session are ordered sequentially, they form the "session identification sid of Π^l for that session".

Partnering. The instances (Π^{l_1} and Π^{l_2}) are called partners to each other, once the following norms are satisfied:

- Π^{l_1} and Π^{l_2} are in "accepted states".
- Π^{l_1} and Π^{l_2} share the same sid for "mutual authentication".
- Π^{l_1} and Π^{l_2} are "mutual partners of each other".

Freshness. An instance $\Pi_{DR_i}^{l_1}$ or $\Pi_{SD_j}^{l_3}$ is said to be *fresh*, if the session key SK_{SD_j,DR_i} $(= SK_{DR_i,SD_j})$ between SD_j and DR_i is not disclosed to $\mathcal{A}$ by executing the Reveal(Π^l)

query described in Table 5.2. In a similar way, $\Pi_{DR_i}^{l_1}$ or $\Pi_{GSS}^{l_2}$ is also *fresh*, if the secret key $SK_{DR_i,GSS}$ $(= SK_{GSS,DR_i})$ between DR_i and GSS is not disclosed to $\mathcal{A}$ by executing the Reveal(Π^l) query described in Table 5.2.

Before going to Theorem 5.1, we define "semantic security" of our proposed AKMS-AgriIoT in Definition 5.1.

Definition 5.1 (Semantic security)**.** *If $Adv_{\mathcal{A}}^{AKMS-AgriIoT}(t_p)$ denote the "advantage of an adversary $\mathcal{A}$ running in polynomial time t_p" to break the semantic security of the proposed AKMS-AgriIoT for computing the established session key SK_{SD_j,DR_i} $(= SK_{DR_i,SD_j})$ between SD_j and DR_i, and secret key $SK_{DR_i,GSS}$ $(= SK_{GSS,DR_i})$ between DR_i and GSS in a particular session. Then,*

$$Adv_{\mathcal{A}}^{AKMS-AgriIoT}(t_p) = |2Pr[c' = c] - 1|,$$

where c and c' are respectively the "correct" and "guessed" bits.

Theorem 5.1. *Let q_h, $|\mathcal{H}|$, and $Adv_{\mathcal{A}}^{ECDDHP}(t_p)$ be the number of "Hash queries", the range space of "a one-way collision-resistant hash function $h(\cdot)$", and the advantage of breaking the "Elliptic Curve Decisional Diffie-Hellman Problem (ECDDHP)", respectively. If an adversary $\mathcal{A}$ running in polynomial time t_p wants to compute session key SK_{SD_j,DR_i} $(= SK_{DR_i,SD_j})$ between SD_j and DR_i, and secret key $SK_{DR_i,GSS}$ $(= SK_{GSS,DR_i})$ between DR_i and GSS in a particular session, then*

$$Adv_{\mathcal{A}}^{AKMS-AgriIoT}(t_p) \leq \frac{q_h^2}{|\mathcal{H}|} + 2Adv_{\mathcal{A}}^{ECDDHP}(t_p).$$

Proof. The proof of this theorem is followed in a similar way that was done in [114, 295, 296, 331, 334]. Here, we adapt the three games, say $\mathbb{G}ame_i$, $i = 0, 1, 2$. Let $Succ_{\mathbb{G}ame_i}$ represent an event of the adversary $\mathcal{A}$'s wining $\mathbb{G}ame_i$ by guessing the correct bit c. We define $\mathcal{A}$'s advantage (success probability) by $Adv_{\mathcal{A},\mathbb{G}ame_i}^{AKMS-AgriIoT} = Pr[Succ_{\mathbb{G}ame_i}]$.

Now, the detailed description of every game $\mathbb{G}ame_i$ is given below.

- $\mathbb{G}ame_0$: Under this game $\mathbb{G}ame_0$, $\mathcal{A}$ executes the real attack against AKMS-AgriIoT under the ROR model. At the beginning of $\mathbb{G}ame_0$, $\mathcal{A}$ picks a random bit c. From the semantic security security defined in Definition 5.1 of the proposed AKMS-AgriIoT, it follows that

$$Adv_{\mathcal{A}}^{AKMS-AgriIoT}(t_p) = |2Adv_{\mathcal{A},\mathbb{G}ame_0}^{AKMS-AgriIoT} - 1|. \tag{5.1}$$

- $\mathbb{G}ame_1$: In this game, $\mathcal{A}$ applies *eavesdropping attack*. $\mathcal{A}$ performs the *Execute* query to intercept all the communicated messages $Msg_{SD_1} = \{TID_{SD_j}, A_{SD_j}, Sign_{SD_j}, TS_{SD_j}\}$ and $Msg_{SD_2} = \{TID^*_{SD_j}, B_{DR_i}, Sign_{DR_{i_1}}, TS_{DR_{i_1}}\}$ during authentication phase (see Section 5.3.3), and also the messages $Msg_{GD1} = \{TID_{DR_i}, X_{GSS}, Cert_{GSS}, Sign_{x_{GSS}}, PID^*_{GSS}, TS_{GSS}\}$ and $Msg_{GD2} = \{TID^*_{DR_i}, Cert_{DR_i}, Y_{DR_i}, Sign_{DR_i}, TS_{DR_i}\}$ during key management phase (see Section 5.3.4) to derive the session keys SK_{SD_j,DR_i} $(= SK_{DR_i,SD_j})$ and $SK_{DR_i,GSS}$ $(= SK_{GSS,DR_i})$. After intercepting all messages, $\mathcal{A}$ can simulate the *Test* and *Reveal* queries to validate whether the derived session keys are correct or just random values. To derive the session keys, $\mathcal{A}$ needs the short term as well as long term secrets. Since $\mathcal{A}$ can not compromise any one of the long term and short term secrets by intercepting any messages Msg_{SD_1}, Msg_{SD_2}, Msg_{GD1}, Msg_{GD2}, the success probability of winning $\mathbb{G}ame_1$ remains unchanged, that is, it is same as that obtained in $\mathbb{G}ame_0$. Hence, it follows that

$$Adv^{AKMS-AgriIoT}_{\mathcal{A},\mathbb{G}ame_1} = Adv^{AKMS-AgriIoT}_{\mathcal{A},\mathbb{G}ame_0}. \tag{5.2}$$

- $\mathbb{G}ame_2$: In this game, $\mathcal{A}$ executes an active attack. The difference between this game and the previous game $\mathbb{G}ame_1$ is that the simulation of *Hash* queries, *CorruptSD*, *CorruptDR* and solving of ECDDHP are included in $\mathbb{G}ame_2$. Assume that $\mathcal{A}$ has all the intercepted messages $\{Msg_{SD_1}, Msg_{SD_2}, Msg_{GD1}, Msg_{GD2}\}$. To derive the session key SK_{SD_i,DR_j} $(= SK_{DR_j,SD_i})$ and the secret key $SK_{DR_i,GSS}$ $(= SK_{GSS,DR_i})$, $\mathcal{A}$ needs to compute $SK_{DR_i,SD_j} = h(DHK_{DR_i,SD_j} \,||Sign_{SD_j} \,||TS_{SD_j} \,||TS_{DR_{i_1}})$ $(= SK_{DR_j,SD_i})$ and $SK_{DR_i,GSS} = h(DHK_{DR_i,GSS} \,||f(PID_{DR_i}, PID_{GSS}) \,||Cert_{DR_i} \,||Cert_{GSS})$ $(= SK_{GSS,DR_i})$. To do so, $\mathcal{A}$ needs to calculate $A_{SD_j} = h(a_{SD_j} \,||TID_{SD_j} \,||PID_{SD_j} \,||k_{SD_j} \,||TS_{SD_j}) \cdot G$, $B_{DR_i} = h(b_{DR_i} \,||TID_{DR_i} \,||PID_{DR_i} \,||k_{DR_i} \,||TS_{DR_{i_1}}) \cdot G$ and $X_{GSS} = h(x_{GSS} \,||k_{GSS} \,||PID_{GSS} \,||TS_{GSS}) \cdot G$ and $Y_{DR_i} = h(y_{DR_i} \,||k_{DR_i} \,||PID_{DR_i} \,||TS_{DR_i}) \cdot G$. Since each value is protected by $h(\cdot)$, to derive the values $\mathcal{A}$ needs to solve ECDDHP also in polynomial time (t_p). Thus, $\mathcal{A}$'s advantage (success probability) of solving ECDDHP in time t_p is $Adv^{ECDDHP}_{\mathcal{A}}(t)$. Again, each message is linked with current timestamp, random nonce (short term secret), and long term secret. If $\mathcal{A}$ simulates the *Hash* queries $\mathcal{H}$ to check the collisions in message digests in the intercepted messages $\{Msg_{SD_1}, Msg_{SD_2}, Msg_{GD1}, Msg_{GD2}\}$, there is a negligible probability of such collisions. Hence, both the games $\mathbb{G}ame_1$ and $\mathbb{G}ame_2$ are indistinguishable if we exclude the simulation of *Hash*, *CorruptSD*, *CorruptDR* queries and solving of ECDDHP. Applying birthday paradox to find the hash collision, the following result is

obtained:

$$|Adv_{\mathcal{A},\mathbb{G}ame_1}^{AKMS-AgriIoT} - Adv_{\mathcal{A},\mathbb{G}ame_2}^{AKMS-AgriIoT}| \leq \frac{q_h^2}{2|\mathcal{H}|} + Adv_{\mathcal{A}}^{ECDDHP}(t_p). \tag{5.3}$$

Once all the queries are executed by $\mathcal{A}$, the only left item is guessing the bit c in order to win the game. It follows that

$$Adv_{\mathcal{A},\mathbb{G}ame_2}^{AKMS-AgriIoT} = \frac{1}{2}. \tag{5.4}$$

Eq. (5.1) gives the semantic security of the proposed AKMS-AgriIoT as

$$\frac{1}{2}.Adv_{\mathcal{A}}^{AKMS-AgriIoT}(t_p) = |Adv_{\mathcal{A},\mathbb{G}ame_0}^{AKMS-AgriIoT} - \frac{1}{2}|. \tag{5.5}$$

Eqs. (5.2), (5.3) and (5.4), and use of triangular inequality lead to the following derivation from Eq. (5.5):

$$\begin{aligned} \frac{1}{2}.Adv_{\mathcal{A}}^{AKMS-AgriIoT}(t_p) &= |Adv_{\mathcal{A},\mathbb{G}ame_0}^{AKMS-AgriIoT} - Adv_{\mathcal{A},\mathbb{G}ame_2}^{AKMS-AgriIoT}| \\ &= |Adv_{\mathcal{A},\mathbb{G}ame_1}^{AKMS-AgriIoT} - Adv_{\mathcal{A},\mathbb{G}ame_2}^{AKMS-AgriIoT}| \\ &\leq \frac{q_h^2}{2|\mathcal{H}|} + Adv_{\mathcal{A}}^{ECDDHP}(t_p). \end{aligned} \tag{5.6}$$

Finally, if we multiply both sides of Eq. (5.6) by "a factor of 2", we arrive at the final result:

$$Adv_{\mathcal{A}}^{AKMS-AgriIoT}(t_p) \leq \frac{q_h^2}{|\mathcal{H}|} + 2Adv_{\mathcal{A}}^{ECDDHP}(t_p).$$

Thus, it is clear that $Adv_{\mathcal{A}}^{AKMS-AgriIoT}(t_p)$ is negligible, because $\frac{q_h^2}{|\mathcal{H}|}$ is negligible and also $Adv_{\mathcal{A}}^{ECDDHP}(t_p)$ is negligible in time t_p. Hence, the theorem follows.

□

5.4.2 Informal security analysis

In the following, we show that the proposed scheme can resist the following attacks.

1) Replay attack

In AKMS-AgriIoT, the current timestamps and the random numbers are utilized in both authentication and key management phases. If an adversary $\mathcal{A}$ tries to replay old messages to the receiving entities, these can be easily detected by means of checking the received

timestamps in the messages with the timestamps when the messages were received by the entities. Therefore, if any replay message contains an old timestamp, it is easily detected by the respective recipient, and that message is simply discarded. AKMS-AgriIoT is then resilient against "replay attack".

2) Man-in-the-middle (MiTM) attack

In this attack, an adversary $\mathcal{A}$ intercepts the authentication request message Msg_{SD_1} between a smart device SD_j and a drone DR_i, and tries to tamper this message to generate another legitimate message, say Msg'_{SD_1}. Without the secret credentials k_{SD_j} and PID_{SD_j}, it is quite infeasible task for $\mathcal{A}$ to generate $A'_{SD_j} = h(a'_{SD_j} \,||TID_{SD_j} \,||PID_{SD_j} \,||k_{SD_j} \,||TS'_{SD_j}) \cdot G$ and signature on a'_{SD_j}, $Sign'_{SD_j}$ for the message $Msg'_{SD_1} = \{TID_{SD_j}, A'_{SD_j}, Sign'_{SD_j}, TS'_{SD_j}\}$, even if $\mathcal{A}$ creates a new random secret a'_{SD_j} and a fresh timestamp TS'_{SD_j}. Similarly, it is also infeasible task for $\mathcal{A}$ to generate valid authentication reply message Msg'_{SD2} between SD_j and DR_i without the secrets k_{DR_i} and PID_{DR_i}, and other messages Msg_{GD_1} and Msg_{GD_2} for the key agreement between DR_i and GSS without secrets PID_{DR_i}, k_{GSS}, PID_{GSS} and k_{DR_i}. Therefore, AKMS-AgriIoT is resilient against MiTM attack.

3) Impersonation attacks

Assume an adversary $\mathcal{A}$ attempts to behave like as a legal smart device SD_j, and constructs an authorized authentication request message $Msg^*_{SD_1}$. To achieve this goal, $\mathcal{A}$ can generate a random secret $a^*_{SD_j}$ and current timestamp $TS^*_{SD_j}$ to calculate valid $A^*_{SD_j}$ and signature $Sign^*_{SD_j}$ on $a^*_{SD_j}$. However, without knowledge of the secret credentials k_{SD_j} and PID_{SD_j}, it is computationally impossible task for $\mathcal{A}$ to create valid $Sign^*_{SD_j}$ and $A^*_{SD_j}$, and consequently, the message $Msg^*_{SD_1} = \{TID_{SD_j}, A^*_{SD_j}, Sign^*_{SD_j}, TS^*_{SD_j}\}$. Therefore, AKMS-AgriIoT is resilient against smart device impersonation attack. In a similar way, it is also computationally expensive to generate legal messages on behalf of a registered drone DR_i and the GSS without having their respective secret credentials. This means that both drone and GSS impersonation attacks are protected in the proposed AKMS-AgriIoT.

4) Privileged-insider attack

During the smart device SD_j, drone (DR_i) , and GSS registration phases, none of the SD_j, DR_i and GSS submits registration credentials to the trusted control room (CR). Instead of that, the CR creates all the credentials, including the secret (private) keys for each entity

prior to their deployment in the IoT environment, and the CR also erases the generated their secret credential from its database. This restricts a "privileged-insider user of the CR, being an insider attacker", can not retrieve any secret information for SD_j, DR_i, and the GSS. AKMS-AgriIoT is then resilient against "privileged-insider attack".

5) Physical IoT smart device and drone capture attacks

According to the discussed threat model in Section 5.1.2, an adversary $\mathcal{A}$ may physically capture some of the smart devices as well as drones. $\mathcal{A}$ can then extract all the stored credentials $\{(TID_{SD_j}, PID_{SD_j}), (k_{SD_j}, Pub_{SD_j})\}$ and $\{(TID_{DR_i}, PID_{DR_i}), f(PID_{DR_i}, y), Cert_{DR_i}, ID_{CR}, (k_{DR_i}, Pub_{DR_i})\}$ from a compromised smart device SD_j and a compromised drone DR_i, respectively, using the "power analysis attacks" [235]. Since the stored secret credentials in each SD_j and DR_i are different as well as unique from the stored information in other smart devices and the drones, compromise of these credentials can not lead to compromise of the session keys established among other non-compromised smart devices and drones too. Therefore, AKMS-AgriIoT is resilient against "physical smart device and drone capture attacks".

6) Ephemeral secret leakage (ESL) attack

In this attack, we apply the CK-adversary model as discussed in the threat model (Section 5.1.2). In authentication phase, a drone DR_i computes the session key SK_{DR_i,SD_j} shared with its associated smart device SD_j as $SK_{DR_i,SD_j} = h(DHK_{DR_i,SD_j} ||Sign_{SD_j} ||TS_{SD_j} ||TS_{DR_{i_1}}) = SK_{SD_j,DR_i}$, where $DHK_{DR_i,SD_j} = DHK_{SD_j,DR_i}$. Now, the computation of DHK_{DR_i,SD_j} involves the random nonces (short term secrets) and private keys (long term secrets) of both DR_i and SD_j. Thus, the session key SK_{DR_i,SD_j} can only be revealed when an adversary $\mathcal{A}$ is in a position to compromise both the ephemeral and long-term secrets. Furthermore, the session keys between all the drones and smart devices are always unique over successive and previous sessions because of current timestamps and random secrets. Even if a session key is known for a particular session, other session keys over other sessions will not be compromised due to the utilization of both short and long-term secrets. In other words, AKMS-AgriIoT is secure against "session-temporary information attack," and it also maintains the "perfect forward and backward secrecy" goals. We can then conclude that AKMS-AgriIoT is resilient against the "ESL attack" under the CK-adversary model.

7) Block verification in blockchain

In AKMS-AgriIoT, assume that a verifier $\mathcal{V}$ wishes to validate a block $Block_m$ stored in the blockchain (as shown in Figure 5.6). In order to do this verification task, $\mathcal{V}$ computes the "Merkle tree root MTR_m^*" on the all encrypted transactions present in that $Block_m$, and also the current block hash, say $CBHash_m^*$ on $Block_m$. If any one of the checks: $MTR_m^* = MTR_m$ and $CBHash_m^* = CBHash_m$ is not valid, $\mathcal{V}$ rejects the $Block_m$. Otherwise, $\mathcal{V}$ further verifies the signature $ECDSA.sig_{Block_m}$ on the transactions using the "ECDSA signature verification algorithm". As a result, a three-level verification process is done to validate a block. In addition, since a block contains the hash value of the previous block, it is quite an impractical task for any adversary to tamper (modify/update) the information stored in the block.

5.5 Formal security verification using AVISPA: simulation study

The "Automated Validation of Internet Security Protocols and Applications (AVISPA)" [9] is one of the popular automated software verification tools that has been used in recent years to verify whether a protocol is "safe", "unsafe" or "inconclusive" against passive/active attacks. Presently, since AVISPA implements only the Dolev-Yao (DY) threat model, it can detect both "replay and man-in-the-middle (MiTM)" attacks. The detailed discussions on AVISPA and its HLPSL implementation are readily available in [9]. The architecture and working of AVISPA have been discussed in detail in Appendix A.

The basic roles are created for CR, SD_j, DR_i, and GSS in the HLPSL language supported by AVISPA. To verify security against replay attack, request and witness functions are applied on the timestamps TS_{SD_j}, TS_{DR_i}, and TS_{GSS} and the private variables a_{SD_j}, b_{DR_i}, x_{GSS}, and y_{DR_i}. In addition, consecutive extra sessions are executed. Two public DY channels are created corresponding to send and receive in each role. Three sessions are created with the intruder impersonating the IoT smart device, the controller node, and the user to verify security against MiTM attack with secrecy goal on random secrets and secret keys, and authentication goal on timestamps and private variables.

In our implementation, we have considered the two cases:

- **Case 1:** authentication phase between drone and smart device (discussed in Section 5.3.3)

- **Case 2:** key management phase between drone and GSS (discussed in Section 5.3.4)

We have then simulated the proposed AKMS-AgriIoT using the OFMC and CL-AtSe backends by considering both the cases under the tool: "SPAN, the Security Protocol ANimator for AVISPA" [22]. The simulation results shown in Figures 5.9 and 5.10 clearly demonstrate that AKMS-AgriIoT is secure against replay and MiTM attacks.

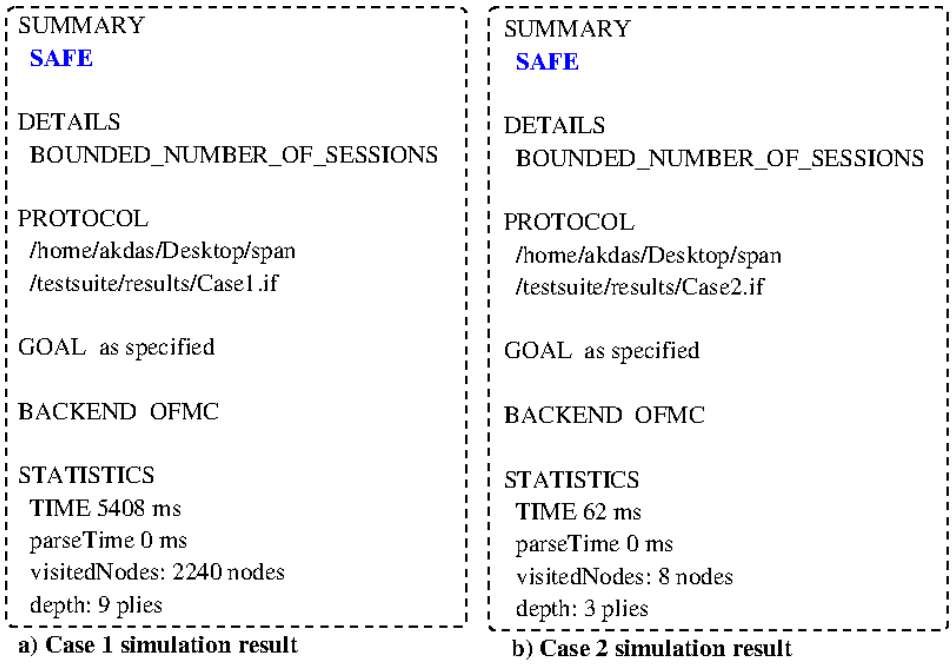

Figure 5.9: Simulation results of AKMS-AgriIoT under OFMC backend for both cases (Case 1 and Case 2)

5.6 Comparative study

This section provides the performance analysis of the proposed AKMS-AgriIoT on communication and computation costs for the authentication phase (Case 1) between a smart device SD_j and a drone DR_i, and the key management phase (Case 2) between a drone DR_i and the GSS as shown in Figure 5.5. In addition, we also provide comparative analysis on communication and computation costs, as well as "security and functionality features" among the proposed AKMS-AgriIoT and other relevant schemes, such as the schemes designed by Tai *et al.* [305], Ali *et al.* [52], Tian *et al.* [311], Sadhukhan *et al.* [274], Shuai *et al.* [286], and Wu and Tsai [340].

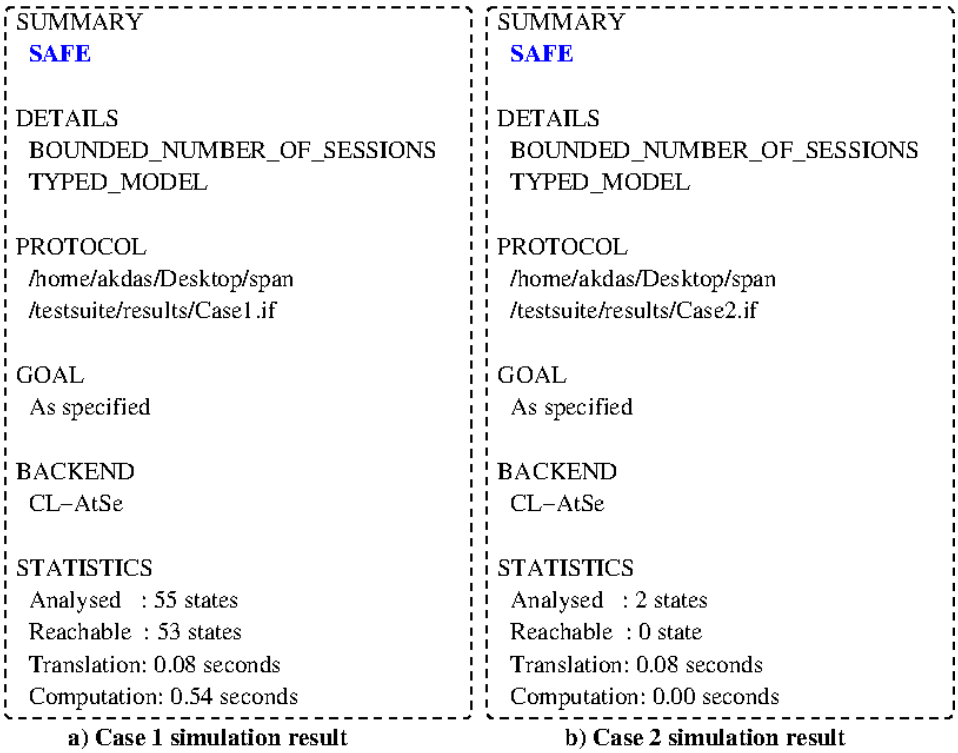

Figure 5.10: Simulation results of AKMS-AgriIoT under CL-AtSe backend for both cases (Case 1 and Case 2)

5.6.1 Security and functionality features comparison

The comparative study on "security and functionality features" (SFA_1–SFA_{15}) on AKMS-AgriIoT and other schemes is finally provided in Table 5.3. It is evident that AKMS-AgriIoT only supports better security features and provides more functionality attributes, such as blockchain solution, dynamic node addition after initial deployment, untraceability and anonymity properties, as compared to the schemes of Tai *et al.* [305], Ali *et al.* [52], Tian *et al.* [311], Sadhukhan *et al.* [274], Shuai *et al.* [286] and Wu and Tsai [340].

5.6.2 Communication costs comparison

For communication cost analysis, the "identity", "random number (nonce)", "elliptic curve point of the form $P = (P_x, P_y)$ where P_x and P_y are x and y coordinates of P respectively", "hash output (if SHA-256 hash algorithm is applied)", and "timestamp" are 160, 160, (160 + 160) = 320, 256 and 32 bits, respectively. In AKMS-AgriIoT, the communication cost for the messages in Case 1, $Msg_{SD_1} = \{TID_{SD_j}, A_{SD_j}, Sign_{SD_j}, TS_{SD_j}\}$,

Table 5.3: Comparison of security & functionality attributes

Attribute	Ali *et al.*	Tian *et al.*	Tai *et al.*	Sadhukhan *et al.*	Shuai *et al.*	Wu and Tsai	AKMS-AgriIoT
SFA_1	✓	✓	✓	✓	✓	✓	✓
SFA_2	×	✓	×	×	×	✓	✓
SFA_3	×	×	×	×	✓	✓	✓
SFA_4	×	×	×	×	✓	✓	✓
SFA_5	✓	✓	✓	✓	✓	✓	✓
SFA_6	✓	✓	×	×	×	×	✓
SFA_7	✓	✓	×	×	×	NA	✓
SFA_8	✓	×	✓	✓	✓	✓	✓
SFA_9	×	×	×	×	×	×	✓
SFA_{10}	✓	✓	✓	✓	✓	✓	✓
SFA_{11}	✓	✓	×	×	×	×	✓
SFA_{12}	✓	N/A	N/A	×	×	×	✓
SFA_{13}	✓	×	✓	✓	✓	✓	✓
SFA_{14}	×	×	×	×	×	✓	✓
SFA_{15}	✓	✓	✓	×	✓	×	✓

✓: "a scheme supports an attribute or resists an attack"; ×: "a scheme does not support an attribute or it does not resist an attack"; N/A: "not applicable in a scheme".
SFA_1: replay attack; SFA_2: privileged insider attack; SFA_3: anonymity; SFA_4: traceability; SFA_5: key agreement; SFA_6: malicious device deployment attack; SFA_7: smart device or drone capture attack; SFA_8: mutual authentication; SFA_9: ESL attack under CK-adversary model; SFA_{10}: man-in-the-middle attack; SFA_{11}: device/drone impersonation attack; SFA_{12}: GSS/server impersonation attack; SFA_{13}: formal security verification under AVISPA tool; SFA_{14}: support to blockchain solution; SFA_{15}: support to dynamic node addition phase.

$Msg_{SD_2} = \{TID^*_{SD_j}, B_{DR_i}, Sign_{DR_{i_1}}, TS_{DR_{i_1}}\}$ demand (160 + 320 + 160 + 32) = 672 bits and (256 + 320 + 160 + 32) = 768 bits respectively, which altogether need 1440 bits. Similarly, the communication cost for Case 2 due to two messages $Msg_{GD1} = \{TID_{DR_i}, X_{GSS}, Cert_{GSS}, Sign_{x_{GSS}}, PID^*_{GSS}, TS_{GSS}\}$, $Msg_{GD2} = \{TID^*_{DR_i}, Cert_{DR_i}, Y_{DR_i}, Sign_{DR_i}, TS_{DR_i}\}$ need (160 + 320 + 160 + 160 + 256 + 32) = 1088 bits and (256 + 160 + 320 + 160 + 32) = 928 bits, respectively, which altogether need 2016 bits. The comparative analysis on communication costs in terms of number of messages and bits required for transmitting the messages among the considered schemes in Table 5.4 exhibits that AKMS-

Table 5.4: Comparison of communication costs

Protocol	No. of messages	Total cost (in bits)
Tai *et al.* [305]	4	2560
Ali *et al.* [52]	5	5504
Tian *et al.* [311]	2	$384s + 11712$
Sadhukhan *et al.* [274]	4	5248
Shuai *et al.* [286]	4	7616
Wu and Tsai [340]	10	$1344+256n$
AKMS-AgriIoT (Case-1)	2	1440
AKMS-AgriIoT (Case-2)	2	2016

Note: s: "number of pseudonyms of a drone in Tian *et al.*'s scheme" [311]; n: "number of agricultural equipment (sensor devices) and blockchains in Wu and Tsai's scheme" [340]

AgriIoT requires low communication costs as compared to other schemes for both the cases (Case 1 and Case 2).

5.6.3 Computation costs comparison

The computation cost analysis for the authentication phase (Case 1) and key management phase (Case 2), we denote T_{poly} as the time required for a t-degree uni-variate polynomial evaluation over $GF(q)$. Furthermore, if we apply the Horner's rule [197], the "evaluation of a t-degree uni-variate polynomial needs t modular multiplications and t modular additions, that is, $T_{poly} = t(T_{mul} + T_{add})$. We consider $t = 100$ in AKMS-AgriIoT".

We utilize the experiment results provided in Appendix B in Table B.1 using MIRACL for the GSS or a server side. On the other side, we utilize the experiment results provided in Table B.2 using MIRACL for a smart device or drone side. In both cases, we use the average time for calculating the computation costs for the proposed AKMS-AgriIoT and other schemes. In Ali *et al.*'s scheme, the fuzzy extractor technique [124] has been applied for biometric verification. We denote T_{fe} as the time required for the fuzzy extractor function. It is assumed that $T_{fe} \approx T_{ecm}$. The comparative study provided in Table 5.5 shows that the proposed AKMS-AgriIoT requires more computation costs for the cases (Case 1 and Case 2) as compared to those for other schemes. This is justified because other schemes are based on lightweight primitives, whereas AKMS-AgriIoT relies on ECC due to support to the

Table 5.5: Comparison of computation costs

Protocol	Smart device/Drone end	GSS/Server end
Ali *et al.* [52]	$11T_h + T_{fe} + 3T_{senc}/T_{sdec}$ ≈ 5.735 ms	$8T_h + 5T_{senc}/T_{sdec}$ ≈ 0.445 ms
Tian *et al.* [311]	$8T_{exp} + 9T_h \approx 4.605$ ms	–
Tai *et al.* [305]	$17T_h \approx 5.253$ ms	$6T_h \approx 0.330$ ms
Sadhukhan *et al.* [274]	$4T_h + 2T_{enc} + 2T_{dec} + 2T_{ecm}$ ≈ 5.876 ms	$2T_h + 2T_{dec} + 2T_{enc}$ ≈ 0.114 ms
Shuai *et al.* [286]	$13T_h + 3T_{exp}$ ≈ 4.701 ms	$7T_h + T_{exp}$ ≈ 0.457 ms
Wu and Tsai [340]	$2T_{bp} + 2T_{exp} + 2T_{enc/dec} + T_h$ ≈ 64.965 ms	$2T_{bp} + 2T_{exp} + 2T_{enc/dec} + T_h$ ≈ 9.407 ms
AKMS-AgriIoT (Case 1)	$11T_h + 8T_{ecm} + 2T_{eca}$ ≈ 21.735 ms	–
AKMS-AgriIoT (Case 2)	$7T_h + 6T_{ecm} + 2T_{eca} + T_{poly}$ ≈ 18.023 ms	$7T_h + 6T_{ecm} + 2T_{eca} + T_{poly}$ ≈ 4.733 ms

blockchain service. However, AKMS-AgriIoT requires significantly low communication costs and provides better security and functionality features provided in Table 5.3 as compared to all other compared schemes.

5.7 Blockchain implementation

This section gives the blockchain implementation of the proposed scheme (AKMS-AgriIoT). During the simulation, if the number of transactions reaches a pre-defined transaction threshold (n_t), we select a leader L from the Peer-to-Peer (P2P) cloud servers (CS) network in a

round-robin fashion for blocks creation, verification as well as addition into the blockchain. With the help of the voting-based PBFT consensus algorithm provided in Algorithms 1 and 2, the leader L adds a block, say $Block_m$ as shown in Figure 5.6, into the blockchain.

Table 5.6: Performance metrics

Characteristics	Consensus algorithm (PBFT)
Byzantine fault tolerance	33%
Crash fault tolerance	33%
Verification speed (transactions per millisecond)	70-80 ms
Message complexity	$\mathcal{O}(n^2)$
Execution complexity	$6T_h + 12T_{ecm} + 6T_{eca}$

The simulation was performed on a platform having the setting: "Ubuntu 18.04, 64-bit OS with Intel(R) Core(TM) i5-4210U CPU @ 1.70GHz, 4 GB RAM". The scripting (programming language) was written in Node.js language with the VS CODE 2019 [191]. We have calculated a block size shown in Figure 5.6 using the following: block version ($BVer_m$), previous block hash (PBH_m), Merkle tree root (MTR_m), timestamp (epoch time) (TS_m), owner of the block (OB_m), public key of signer (Pub_{GSS}), encrypted transaction ($E_{Pub_{GSS}}(Tx_i)$), current block hash (using SHA-256 hashing algorithm) ($CBHash_m$), and ECDSA signature ($ECDSA.sig_{Block_m}$) are of the sizes 32 bits, 256 bits, 256 bits, 32 bits, 160 bits, 320 bits, 640 bits, 256 bits, and 320 bits, respectively. Moreover, each transaction Tx_w was encrypted using ECC encryption which outputs two elliptic curve points, and as a result, an encrypted transaction requires (320+320) = 640 bits. Therefore, the total block size of a block $Block_m$ turns out to be $1642 + 640n_t$ bits. Moreover, in Table 5.6, we have listed the performance measures relevant to the voting-based PBFT consensus algorithm. Note that Algorithms 1 and 2 require four sub-phases, namely pre-prepare, prepared, commit and reply, where in each round n^2 messages are communicated. Therefore, the total number of messages required in Algorithms 1 and 2 is $4n^2 = O(n^2)$, where n is the total number of P2P nodes.

The details of the simulations are provided in three scenarios, as shown in Figures 5.11, 5.12 and 5.13. We have considered the total number of peer nodes in the P2P network as 13.

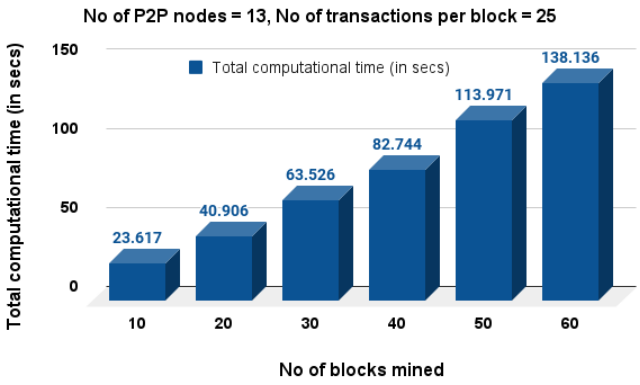

Figure 5.11: Blockchain simulation results for Scenario 1

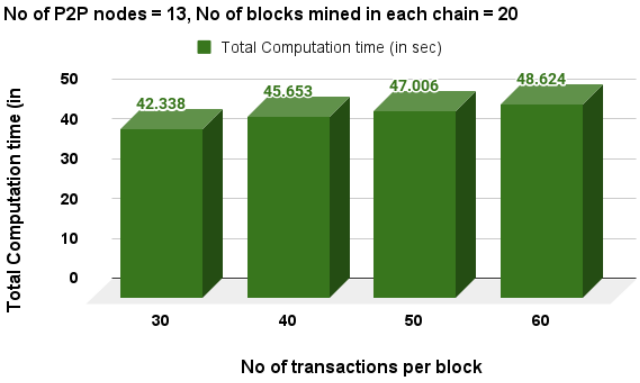

Figure 5.12: Blockchain simulation results for Scenario 2

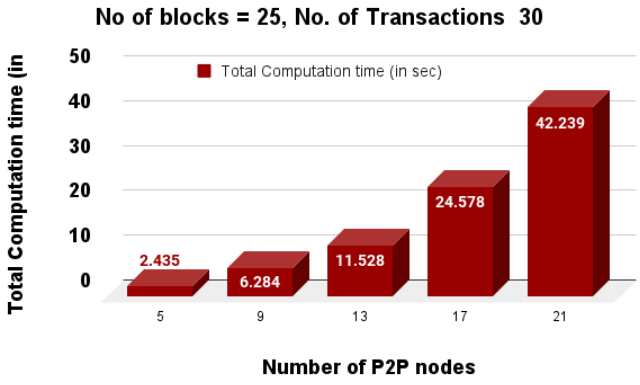

Figure 5.13: Blockchain simulation results for Scenario 3

- **Scenario 1:** This scenario considers the number of transactions per block as 25. The simulation results provided in Figure 5.11 show that if the number of blocks mined is increased, the total computational time also increases linearly.
- **Scenario 2:** This scenario also considers the number of mined blocks in each chain is 20. The simulation results demonstrated in Figure 5.12 illustrate that the total computational time increases linearly when the number of transactions per block increases during the consensus process.
- **Scenario 3:** In this scenario, we consider the number of blocks to be 25, and the number of transactions is 30 under different P2P nodes in the virtual distributed system. The simulation results are shown in Figure 5.13. When the number of P2P nodes is varied, the total computational time (in milliseconds) also increases linearly.

5.8 Summary

This work is an attempt to propose an interesting security scheme, namely the blockchain-based authentication and key agreement in an IoT-enabled agriculture environment using drones, called AKMS-AgriIoT. The data are securely gathered by the GSS from the drones, whereas the drones also collect data securely from their respective deployed IoT smart devices in flying zones. After transactions are formed with the secured collected data, the GSS sends the list of encrypted transactions along with their signatures to its associated cloud server in the blockchain center (BC). The cloud server is then responsible for mining the blocks using the PBFT consensus algorithm to verify and add them to the BC. A detailed security analysis using the formal security verification under the AVISPA tool reveals that AKMS-AgriIoT can resist various potential attacks needed in an IoT environment. Moreover, the comparative study also reveals that AKMS-AgriIoT provides a better trade-off between "security and functionality features" and "communication and computation overheads" as compared to other schemes.

Chapter 6

Smart Contract-Based Blockchain-Envisioned Authentication Scheme for Smart Farming

A smart farming network based on blockchain technology instantly provides agricultural data to the farmers on a single integrated platform. In the agriculture backdrop, certain data belonging to the production process may be private to a particular stakeholder. In contrast, certain other data must be made available to some or all stakeholders to maintain transparency. Data such as the trading transaction details between farmers and agricultural firms must be kept private from the third parties. Other data, such as the chemicals used in the preservation of produce and the amount and quality of fertilizers used in growing the crops, may be made available to the end-product buyers to decide which of the available products to purchase. This implies that some data needs to be encrypted while other data is kept unencrypted.

Private blockchains can be used to store the classified data that is essential to a part of the stakeholder community. Data considered confidential to those specific stakeholders must be encrypted before being placed permanently on the blockchain. Chapter 5 presented a mutual authentication scheme for storing such sensitive data. To store farming data relevant to all stakeholders, a hybrid blockchain is more suitable as it can include both openly available data and confidential data. Hybrid blockchain in particular has the ability to keep both sensitive data and open-access data at the same place allowing better application development for organizations related to smart farming. It is permissioned in requiring a central authority to decide joining of new nodes and permissionless in limiting the privileges of already joined

nodes. Thus, a set of validator nodes are chosen by a central authority to perform blockchain operations. Also, the single point of failure possible at the ground station server in the architecture in the network model presented in Section 5.1.1 is eliminated in the proposed model in this chapter.

Smart contracts are highly relevant to the smart agriculture system with no third party to verify if the agreed terms are followed by the untrustworthy communicating entities within the system. The organizational rules and conditions to follow a protocol for a contractual agreement are converted into programmable codes and placed on the blockchain. Due to the immutability property, these codes cannot be tampered with by anyone with unjust interests [260]. This chapter explores the usage of hybrid blockchains leveraging smart contracts to design a mutual authentication scheme.

6.1 System models

This section provides a brief discussion on the applied network and threat models that are utilized in designing and analyzing the proposed scheme.

6.1.1 Network model

Blockchain-envisioned smart farming architecture shown in Figure 6.1 consists of multiple agricultural fields. Each field may be divided into two or more farm zones with IoT smart devices placed in each zone capable of extracting environment readings from its zone. Each field is assigned to a gateway node that collects readings from the smart devices of all its zones. The data from a smart device may be relayed to the gateway node on a hop-by-hop forwarding basis. The gateway node has collected transactions of sensor data as well as other data such as the trading transaction details between farmers and agricultural firms, the chemicals used in the preservation of produce, the amount and quality of fertilizers used in growing the crops, economic gain of usable produce and quantity of unusable produce. These transactions are then forwarded to the edge server by the gateway node for initial processing and creation of partial blocks after deciding which of the transactions in the received block should be encrypted and which can be maintained unencrypted to contribute to the hybrid blockchain. The initially processed data in the form of partial blocks are then sent to the blockchain cloud center consisting of several cloud servers with smart contracts facility. The cloud server(s) first use the smart contracts to validate the received partial block. Once it is

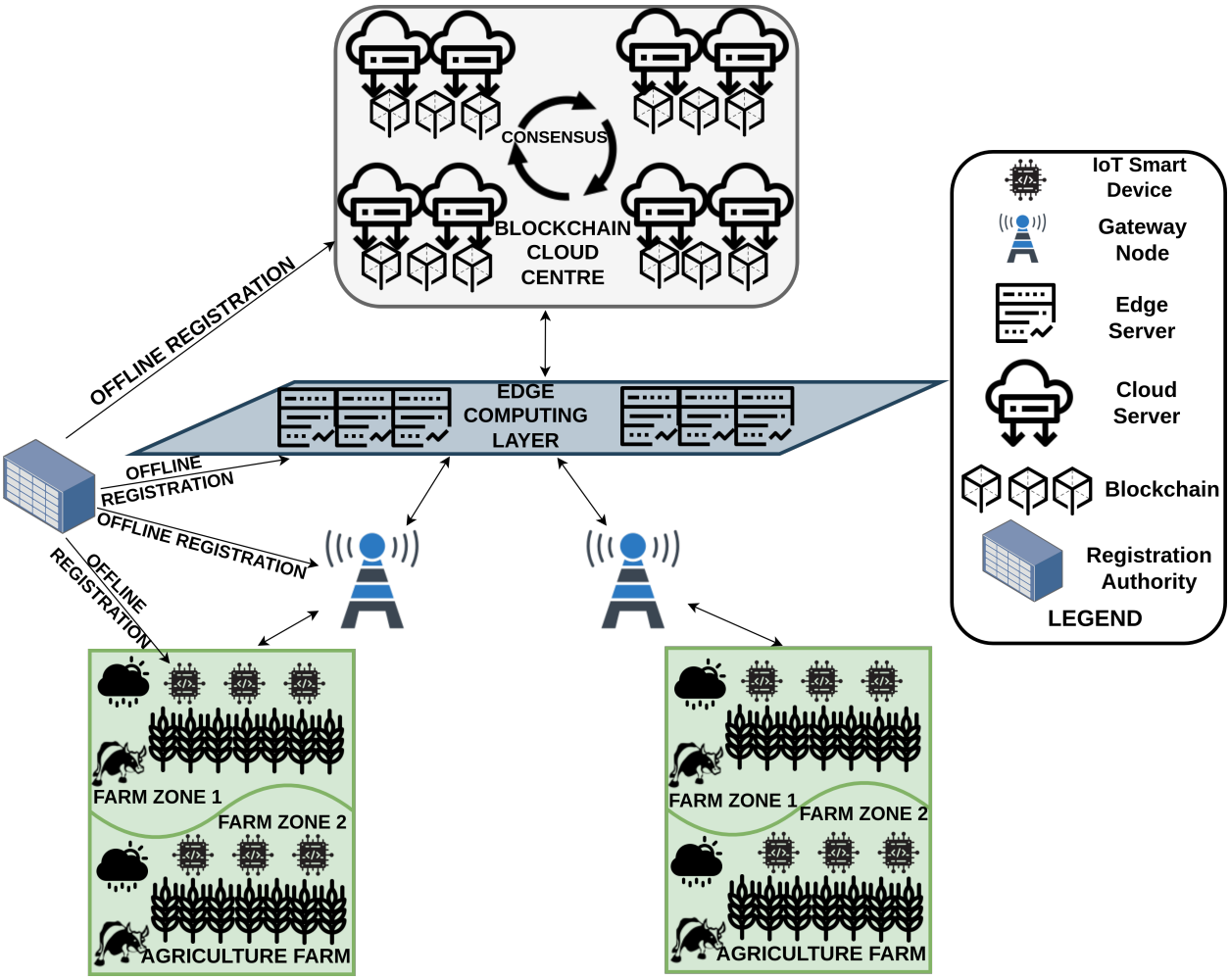

Figure 6.1: Blockchain-envisioned smart farming architecture

validated, the cloud server creates the full block, executes a consensus algorithm with other cloud servers, and then performs the addition of the full block into the blockchain.

6.1.2 Threat model

The threat models "Dolev-Yao (DY) threat model" [125] along with the contemporary *de facto* "Canetti and Krawczyk's model (CK-adversary model)" [94] discussed in Section 1.4.1 of Chapter 1 are applied. The source and destination entities (e.g., IoT smart devices) cannot be trusted. In addition, it is assumed that the gateway nodes, edge servers, and cloud servers are semi-trusted entities. A is assumed to have the following capabilities:

- $\mathcal{A}$ can seize, remove, modify and re-transmit existing messages or circulate counterfeit

messages for any communication in public channels between IoT smart devices, gateway nodes, edge servers, and cloud servers.

- Impersonation of IoT smart devices and gateway nodes may be carried out by $\mathcal{A}$ to perform actions on their behalf.
- Simultaneous multiple executions of the protocol may be initialized by $\mathcal{A}$. The IoT smart devices, gateway nodes, edge servers, and cloud servers may take part in any number of such concurrent executions at the same time.
- The IoT smart devices, gateway nodes, edge servers, and cloud servers are honest and stateless. $\mathcal{A}$ is stateful.
- Hijacking of session states during communication among IoT smart devices, gateway nodes, edge servers and cloud servers may be employed by $\mathcal{A}$ to extract secret credentials.

In addition, physical capture of IoT smart devices is employed by $\mathcal{A}$ using power analysis attacks [200, 235] and timing attacks [199] to extract secret credentials from their memory. The adversary cannot compromise the gateway nodes, edge servers, and cloud servers since they are put under a physical locking system as suggested in [75, 335]. The secret credentials in the gateway nodes, edge servers, and cloud servers will be stored in their secure databases so that the stolen verifier attack is prevented. Thus, the adversary will have no access to the credentials stored in the IoT smart devices, gateway nodes, edge servers, and cloud servers through the stolen verifier attack so that no other attacks, such as IoT smart devices, gateway nodes, edge servers, and cloud servers impersonation attack can be launched through the stolen verifier attacks.

6.2 Research contributions

The main research contributions are many-fold:

1) Smart contract-based blockchain-envisioned authenticated key agreement mechanism in a smart farming environment. The device-to-device (D2D) authentication phase and device-to-gateway (D2G) authentication phase support mutual authentication and key agreement between two Internet of Things (IoT) enabled devices and between an IoT device and the gateway node in the network, respectively.

2) The formed blocks contain the encrypted/unencrypted sensing data from the IoT smart devices as well as other data such as the trading transaction details between farmers and agricultural firms and the chemicals used in the preservation of produce, the amount and quality of fertilizers used in growing the crops may be made available to the buyers of end-product for deciding among which of the available products to purchase. Thus, a hybrid blockchain is used in the proposed scheme. The voting-based "Practical Byzantine Fault Tolerance (PBFT)" [95] consensus algorithm has been applied with the help of smart contracts to validate and add the blocks into the blockchain using a P2P cloud server network.

3) A detailed security analysis using formal, informal, and formal security verification with an automated software validation tool, known as "Automated Validation of Internet Security Protocols and Applications (AVISPA)" [9] has been carried out to show the robustness of the proposed scheme against several potential attacks.

4) A testbed experiment has been done for measuring the computational time needed for various cryptographic primitives under the "MIRACL Cryptographic SDK: Multiprecision Integer and Rational Arithmetic Cryptographic Library" [38] under both a server and a Raspberry PI 3 settings. Next, a detailed comparative analysis shows the proposed scheme provides superior security and more functionality features and requires low communication costs and comparable computation costs compared to the existing competing authentication schemes.

5) A blockchain-based implementation has also been carried out for measuring computational time for a varied number of blocks and also a varied number of transactions per block.

6.3 The proposed hybrid blockchain-based scheme

To ensure that the data exchanged between the components come from the expected reliable source, an authentication scheme is designed to be executed during the hop-by-hop device data relay between any two IoT smart devices that are forwarding the data, and another authentication scheme is designed to be executed when a smart device forwards its data to the associated gateway node. A gateway node uses the public key of the edge server to encrypt the device data when forwarding to an edge server. Similarly, an edge server uses the public key of the cloud server to encrypt the device data when forwarding it to a

cloud server. The edge server between the gateway and the cloud server is used to decide which transactions in the received block should be encrypted and which can be maintained unencrypted to generate the hybrid blockchain.

For replay attack protection, both the random numbers (secrets) and current timestamps generated by the respective entities in the network are considered. Thus, we assume that various entities (IoT smart devices and the gateway nodes) are synchronized with their clocks, which is a typical assumption applied in many recent authentication and access control protocols in IoT deployment [74, 100, 114, 127, 151, 227, 270, 333, 335]. We utilize the notations in the proposed smart contract-based blockchain-envisioned authentication scheme in the smart farming environment (SCBAS-SF) provided in Table 6.1.

6.3.1 System initialization phase

In this phase, a trusted registration authority (RA) selects the system parameters using the following steps:

Step SIP_1: The RA selects a non-singular elliptic curve $E_q(a, b) : y^2 = x^3 + ax + b \pmod q$ over the Galois field $GF(q)$, with a "point at infinity (zero point)" $\mathcal{O}$, constants $a, b \in Z_q = \{0, 1, 2, \cdots, q-1\}$ such that $4a^3 + 27b^2 \neq 0 \pmod q$ is satisfied. The RA picks a base point $G \in E_q(a, b)$ whose order n_G is as large as q, that is, $n_G \cdot G = G+G+ \cdots +G$ (n_G times) $= \mathcal{O}$ [198].

Step SIP_2: The RA picks a "collision-resistant one-way cryptographic hash function", say $H(\cdot)$ (for instance, SHA-256 hash algorithm may be used [232]) and the "Practical Byzantine Fault Tolerance (PBFT)" algorithm to be used in consensus process in the blockchain cloud center.

Step SIP_3: Finally, the RA picks its own master key $mk_{RA} \in Z_q^*$ and publishes the domain parameters $\{E_q(a, b), G, H(\cdot)\}$ as public.

6.3.2 Registration phase

In this phase, we discuss various registration processes of the entities involved in the network.

1) **Registration of IoT smart devices:** This phase is executed offline by the RA to register the IoT smart devices (SN) in a particular zone in the agriculture field. The following steps are essential to complete IoT smart devices registration process:

Table 6.1: Notations and their description

Notation	Significance
$E_q(a,b)$	A non-singular elliptic curve of the form: $y^2 = x^3 + ax + b \pmod q$
G	A base point in $E_q(a,b)$ of order n_G as big as q
$x \cdot G$	An elliptic curve point multiplication: $x \cdot G = G + G + \cdots + G\,(x \text{ times})$
$P + Q$	Elliptic curve point addition; $P, Q \in E_q(a,b)$
RA	Registration Authority
GWN	Gateway node
SN	IoT smart (sensor) device
ES	Edge server
CS	Cloud server
RTS_X	Registration timestamp issued by the RA to an entity X
TC_S	SN's temporal credential
ID_X, TID_X, RID_X	Entity X's real, temporary and pseudo identities, respectively
mk_{RA}	Master key of RA
pr_X, Pub_X	Private and public keys of entity X, respectively
s_1, g_2, e_1, c_1	RA's random secrets
r_{S1}, r_{S2}, p_S	SN's random secrets
q_G	GWN's random secrets
$\|\|$	Concatenation operation
TS_X	Current timestamp generated by an entity X
$*$	Modular multiplication in a finite field Z_q
$\oplus$	Exclusive-OR (XOR) operation
ΔT	Maximum transmission delay related to a message
$H(\cdot)$	"Collision-resistant cryptographic one-way hash function"

- *Step* $SDRE_1$: The RA picks a real identity ID_S and temporary identity TID_S, and a random secret $s_1 \in Z_q^*$ to compute a pseudo-identity for SN as $RID_S = H(ID_S||s_1||mk_{RA})$.
- *Step* $SDRE_2$: The RA picks a random private key $pr_S \in Z_q^*$, and computes the

corresponding public key $Pub_S = pr_S \cdot G$ and a temporal credential for the SN as $TC_S = H(RID_S || pr_S || mk_{RA} || RTS_S)$ where RTS_S is the current timestamp of registration of SN.

- *Step* $SDRE_3$: The RA preloads SN with the credentials $\{(RID_S, TID_S, TC_S), H(\cdot), E_q(a,b), G, (pr_S, Pub_S)\}$. In addition, the RA publishes Pub_S as SN's public key.

The summary of this phase is illustrated in Figure 6.2.

Trusted Registration Authority (RA)	IoT Smart Device (SN)
Pick ID_S, TID_S, $s_1 \in Z_q^*$	
Compute $RID_S = H(ID_S \|\| s_1 \| mk_{RA})$	
Pick $pr_S \in Z_q^*$	
Compute $Pub_S = pr_S \cdot G$,	
$TC_S = H(RID_S \|\| pr_S \|\| mk_{RA} \|\| RTS_S)$.	
Preload SN with	Store $\{(RID_S, TID_S, TC_S)$,
$\{(RID_S, TID_S, TC_S), H(\cdot)$,	$H(\cdot), E_q(a,b), G, (pr_S, Pub_S)\}$
$E_q(a,b), G, (pr_S, Pub_S)\}$	in its memory

Figure 6.2: Summary of IoT smart device registration phase

2) **Registration of gateway nodes, edge servers and cloud servers:** In this phase, the gateway nodes (GWN), the edge servers (ES), and the cloud servers (CS) are registered by the RA in offline mode. The association between the IoT smart devices and the gateway node, the edge server and the gateway nodes, the cloud server and the edge server nodes is decided during the deployment. This allows the gateway node to collect data from only its associated smart devices, the edge server to receive data from only its associated gateway nodes, and the cloud server to receive data from only its associated edge servers. This phase consists of the following steps:

- *Step* SR_1: The RA picks the real identity for a gateway node (GWN), an edge server (ES) and a cloud server (CS) as ID_G, ID_E and ID_C, respectively, and their temporary identities as TID_G, TID_E and TID_C, respectively. In addition, the RA picks random secrets g_1, e_1, $c_1 \in Z_q^*$ and computes the pseudo-identities as $RID_G = H(ID_G || g_1 || mk_{RA} || RTS_G)$, $RID_E = H(ID_E || e_1 || mk_{RA} || RTS_E)$ and $RID_C =$

$H(ID_C||\ c_1||\ mk_{RA}||\ RTS_C)$, where RTS_G, RTS_E, and RTS_C are the registration timestamps of GWN, ES and CS, respectively.

- *Step* SR_2: The RA preloads the gateway node GWN with the credentials $\{(RID_G, TID_G), \{(RID_S, TID_S, TC_S)\}, H(\cdot), E_q(a,b), G\}$. After that GWN picks its own random private key $pr_G \in Z_q^*$ and computes the corresponding public key $Pub_G = pr_G \cdot G$. The GWN adds its private and public key (pr_G, Pub_G) to its tamper-proof secure memory database as $\{(RID_G, TID_G), \{(RID_S, TID_S, TC_S)\}, (pr_G, Pub_G), H(\cdot), E_q(a,b), G\}$. Furthermore, the GWN publishes Pub_G as its public key.
- *Step* SR_3: The RA sends the credentials $\{(RID_E, TID_E), \{(RID_G)\}, H(\cdot), E_q(a,b), G\}$ to the edge server ES via secure channel. ES picks its own random private key $pr_E \in Z_q^*$ and computes the corresponding public key $Pub_E = pr_E \cdot G$. ES adds its private and public key (pr_E, Pub_E) to its tamper-proof secure memory database as $\{(RID_E, TID_E), \{(RID_G)\}, (pr_E, Pub_E), H(\cdot), E_q(a,b), G\}$. In addition, ES publishes Pub_E as its public key.
- *Step* SR_4: The RA sends the credentials $\{(RID_C, TID_C), \{(RID_E)\}, H(\cdot), E_q(a,b), G\}$ securely to the cloud server CS. CS then picks its own random private key $pr_C \in Z_q^*$ and computes the corresponding public key $Pub_C = pr_C \cdot G$. CS adds its private and public key (pr_C, Pub_C) to its tamper-proof secure memory database as $\{(RID_C, TID_C), \{(RID_E)\}, (pr_C, Pub_C), H(\cdot), E_q(a,b), G\}$. Finally, CS publishes Pub_C as its public key.

  The registration phase related to GWN, ES and CS is summarized in Figure 6.3.

6.3.3 Authentication and key agreement phase

In this phase, we discuss the device-to-device (D2D) and device-to-gateway node (D2G) authentication phases.

1) **Device-to-device authentication phase:** An IoT Smart Device SN_1 needs to send securely its sensing data to another IoT Smart device SN_2. Since the two smart devices are registered by the RA, the information available in their memory are used in the D2D authentication to mutually authenticate each other and establish a session key between them using the following steps:

   - *Step* $D2D_1$: SN_1 as the initiator first picks a random secret $r_{S1} \in Z_q^*$ and current timestamp TS_{S1} to calculate $x_{SN_1} = H(RID_{S1}||\ TID_{S1}||\ TC_{S1}||\ pr_{S1}||\ TS_{S1}||\ r_{S1})$,

Registration Authority (RA)	Gateway Node(GWN)	Edge Server (ES)	Cloud Server (CS)
Pick ID_G, TID_G, g_1, ID_E, TID_E, e_1, ID_C, TID_C, $c_1 \in Z_q^*$ Compute $RID_G = H(ID_G \|\| g_1 \|\| mk_{RA} \|\| RTS_G)$, $RID_E = H(ID_E \|\| e_1 \|\| mk_{RA} \|\| RTS_E)$, $RID_C = H(ID_C \|\| c_1 \|\| mk_{RA} \|\| RTS_C)$ Preload the gateway node GWN with $\{(RID_G, TID_G), \{(RID_S, TID_S, TC_S)\}, H(\cdot), E_q(a,b), G\}$ Preload ES with $\{(RID_E, TID_E), \{(RID_G)\}, H(\cdot), E_q(a,b), G\}$ Preload CS with $\{(RID_C, TID_C), \{(RID_E)\}, H(\cdot), E_q(a,b), G\}$	Pick $pr_G \in Z_q^*$ Compute $Pub_G = pr_G \cdot G$ Store $\{(RID_G, TID_G), \{(RID_S, TID_S, TC_S)\}, (pr_G, Pub_G), H(\cdot), E_q(a,b), G\}$ in secure database	Pick $pr_E \in Z_q^*$ Compute $Pub_E = pr_E \cdot G$ Store $\{(RID_E, TID_E), \{(RID_G)\}, (pr_E, Pub_E), H(\cdot), E_q(a,b), G\}$ in secure database	Pick $pr_C \in Z_q^*$ Compute $Pub_C = pr_C \cdot G$ Store $\{(RID_C, TID_C), \{(RID_E)\}, (pr_C, Pub_C), H(\cdot), E_q(a,b), G\}$ in secure database

Figure 6.3: Summary of registration phase of GWN, ES and CS

$X_{S1} = x_{S1} \cdot G$, the signature on r_{S1} as $Sig_{S1} = x_{S1} + H(TID_{S1} \|\| Pub_{S1} \|\| TS_{S1}) * pr_{S1} \pmod q$. SN_1 sends the authentication request message $Msg_{D2D_1} = \{TID_{S1}, X_{S1}, Sig_{S1}, Pub_{S1}, TS_{S1}\}$ to its neighbor IoT device SN_2 via open channel.

- *Step D2D*$_2$: SN_2 as the responder receives Msg_{D2D_1} at time TS_{S1}^* and checks the timestamp validity by $|TS_{S1}^* - TS_{S1}| \leq \Delta T$. If it is valid, SN_2 verifies the received signature as $Sig_{S1} \cdot G = X_{S1} + H(TID_{S1} \|\| Pub_{S1} \|\| TS_{S1}) \cdot Pub_{S1}$. If the signature is valid, SN_2 generates a random secret $r_{S2} \in Z_q^*$, and current timestamp TS_{S2} for computing $y_{S2} = H(TID_{S2} \|\| RID_{S2} \|\| TC_{S2} \|\| r_{S2} \|\| pr_{S2} \|\| TS_{S2})$, $Y_{S2} = y_{S2} \cdot G$, the secret (session key) shared with SN_1 as $SK_{S2S1} = y_{S2} \cdot X_{S1}$ and the signature on the session key SK_{S2S1} as $Sig_{S2} = y_{S2} + H(TID_{S2} \|\| TID_{S1} \|\| Pub_{S2} \|\| SK_{S2S1} \|\| TS_{S2}) * pr_{S2} \pmod q$. SN_2 sends the authentication response message $Msg_{D2D_2} = \{TID_{S2}, Y_{S2}, Sig_{S2}, Pub_{S2}, TS_{S2}\}$ to SN_1 via open channel.

- *Step D2D*$_3$: If SN_1 receives Msg_{D2D_2} at time TS_{S2}^*, it validates the timestamp by the condition: $|TS_{S2}^* - TS_{S2}| \leq \Delta T$. If it is valid, SN_1 computes the secret (session key) shared with SN_2 as $SK_{S1S2} = x_{S1} \cdot Y_{S2}$. SN_1 validates the received signature by $Sig_{S2} \cdot G = Y_{S2} + H(TID_{S2} \|\| TID_{S1} \|\| Pub_{S2} \|\| SK_{S1S2} \|\| TS_{S2}) \cdot Pub_{S2}$. SN_1 then creates a new timestamp TS_{S3}, computes the session key verifier as $SKV_{S1S2} = H(SK_{S1S2} \|\| TS_{S3})$, and sends the authentication acknowledgement message $Msg_{D2D_3} = \{SKV_{S1S2}, TS_{S3}\}$ to SN_2 via public channel.

- *Step D2D*$_4$: Let SN_2 receive Msg_{D2D_3} at time TS^*_{S3}. SN_2 validates the timestamp TS_{S3} by $|TS^*_{S3} - TS_{S3}| \leq \Delta T$. If it is valid, SN_2 computes the session key verifier as $SKV_{S2S1} = H(SK_{S2S1}|| \ TS_{S3})$. SN_2 checks if $SKV_{S2S1} = SKV_{S1S2}$. If it is valid, both SN_1 and SN_2 share the same secret key $SK_{S2S1} = SK_{S1S2}$ for their secret communication.

The summary of this scheme is shown in Figure 6.4.

IoT Smart Device (SN_1)	IoT Smart Device (SN_2)
Pick $r_{S1} \in Z^*_q$ and timestamp TS_{S1} Compute $x_{S1} = H(RID_{S1}\|\| \ TID_{S1}\|\| \ TC_{S1}\|\| \ pr_{S1}\|\| \ TS_{S1}\|\| \ r_{S1})$ $X_{S1} = x_{S1} \cdot G$ Calculate $Sig_{S1} = x_{S1}$ + $H(TID_{S1}\|\| \ Pub_{S1}\|\| \ TS_{S1}) * pr_{S1} \pmod q$. $Msg_{D2D_1} = \langle TID_{S1}, X_{S1}, Sig_{S1}, Pub_{S1}, TS_{S1} \rangle \rightarrow$ via public channel	Check if $\|TS^*_{S1} - TS_{S1}\| \leq \Delta T$? If so, check if $Sig_{S1} \cdot G = X_{S1}$ + $H(TID_{S1}\|\| \ Pub_{S1}\|\| \ TS_{S1}) \cdot Pub_{S1}$? If so, generate $r_{S2} \in Z^*_q$, TS_{S2} Compute $y_{S2} = H(TID_{S2}\|\| \ RID_{S2}\|\| \ TC_{S2}\|\| \ r_{S2}\|\| \ pr_{S2}\|\| \ TS_{S2})$ $Y_{S2} = y_{S2} \cdot G$ Compute $SK_{S2S1} = y_{S2} \cdot X_{S1}$ Compute $Sig_{S2} = y_{S2}$ + $H(TID_{S2}\|\| \ TID_{S1}\|\| \ Pub_{S2}\|\| \ SK_{S2S1}\|\| \ TS_{S2}) * pr_{S2} \pmod q$ $Msg_{D2D_2} = \langle TID_{S2}, Y_{S2}, Sig_{S2}, Pub_{S2}, TS_{S2} \rangle \leftarrow$ via public channel
Check if $\|TS^*_{S2} - TS_{S2}\| \leq \Delta T$? If so, compute $SK_{S1S2} = x_{S1} \cdot Y_{S2}$ $Sig_{S2} \cdot G = Y_{S2}$ + $H(TID_{S2}\|\| \ TID_{S1}\|\| \ Pub_{S2}\|\| \ SK_{S1S2}\|\| \ TS_{S2}) \cdot Pub_{S2}$ Generate TS_{S3} Compute $SKV_{S1S2} = H(SK_{S1S2}\|\| \ TS_{S3})$ $Msg_{D2D_3} = \langle SKV_{S1S2}, TS_{S3} \rangle \rightarrow$ via public channel	
	Check if $\|TS^*_{S3} - TS_{S3}\| \leq \Delta T$? If so, compute $SKV_{S2S1} = H(SK_{S2S1}\|\| \ TS_{S3})$ Check $SKV_{S2S1} \stackrel{?}{=} SKV_{S1S2}$ If so, store SKV_{S2S1}
Store SK_{S1S2}	
Both SN_1 and SN_2 share the same secret key $SK_{S2S1} = SK_{S1S2}$	

Figure 6.4: Summary of D2D authentication phase

2) **Device-to-gateway authentication phase:** In this phase, an IoT smart device SN and its associated gateway node GWN mutually authenticate each other and establish a secret session key among them with the help of the following steps:

- *Step D2G*$_1$: SN picks a random secret $p_S \in Z^*_q$ and current timestamp TS_S to calculate $A_S = H(TID_S|| \ p_S|| \ TS_S) \cdot G$, $x_S = H(pr_S|| \ p_S|| \ RID_S|| \ TS_S) \oplus H(TC_S|| \ TID_S|| \ RID_S|| \ TS_S)$ and the signature on p_S as $Sig_S = H(TID_S|| \ p_S|| \ TS_S)$ + $H(x_S|| \ TID_S|| \ A_S|| \ TS_S|| \ TC_S) * pr_S \pmod q$. It then sends the authentica-

tion request message $Msg_{D2G_1} = \langle TID_S, A_S, x_S, Sig_S, TS_S \rangle$ to the GWN via open channel.

- *Step D2G*$_2$: The GWN receives the request message Msg_{D2G_1} from SN at time TS_S^* and validates the timestamp by $|TS_S^* - TS_S| \leq \Delta T$. If it is valid, the received signature is verified by $Sig_S \cdot G = A_S + H(x_S || TID_S || A_S || TS_S || TC_S) \cdot Pub_S$. If the signature is valid, the GWN computes $y_S = H(pr_S || p_S || RID_S || TS_S) = x_S \oplus H(TC_S || TID_S || RID_S || TS_S)$. Furthermore, the GWN generates a random secret $q_G \in Z_q^*$ and current timestamp TS_G to calculate $B_G = H(TID_G || q_G || RID_S || TS_G) \cdot G$, $DK_{GS} = H(TID_G || q_G || RID_S || TS_G) \cdot A_S$, $y_G = H(pr_G || TID_G || RID_G || TC_S || TS_G) \oplus H(TID_G || TC_S || TS_S || TS_G || RID_S)$, and the session key shared with SN as $SK_{GS} = H(DK_{GS} || y_S || H(pr_G || TID_G || RID_G || TC_S || TS_G))$. GWN also generates a new temporary identity for SN as TID_S^{new} and encrypts it as $TID_S^* = TID_S^{new} \oplus H(SK_{GS} || TS_G)$. It then creates a signature on q_G as $Sig_G = H(TID_G || q_G || RID_S || TS_G) + H(TID_S || TID_G || TC_S || y_G || Pub_G || TS_G) * pr_G \pmod q$ and sends the authentication response message $Msg_{D2G_2} = \langle TID_G, B_G, y_G, Sig_G, TID_S^*, TS_G \rangle$ to SN via open channel.

- *Step D2G*$_3$: After receiving Msg_{D2G_2} at TS_G^*, SN verifies the timestamp TS_G by $|TS_G^* - TS_G| \leq \Delta T$. If it is deemed to be valid, SN proceeds to calculate $DK_{SG} = H(TID_S || p_S || TS_S) \cdot B_G$, $z_G = H(pr_G || TID_G || RID_G || TC_S || TS_G) = y_G \oplus H(TID_G || TC_S || TS_S || TS_G || RID_S)$, and the session key shared with the GWN as $SK_{SG} = H(DK_{SG} || H(pr_S || p_S || RID_S || TS_S) || z_G)$. SN verifies the received GWN's signature as $Sig_G \cdot G = B_G + H(TID_S || TID_G || TC_S || y_G || Pub_G || TS_G) \cdot Pub_G$. If the signature is valid, SN extracts $TID_S^{new} = TID_S^* \oplus H(SK_{GS} || TS_G)$ and updates TID_S with TID_S^{new} in its memory. It also generates a current timestamp TS_V to compute the session key verifier as $SKV_{SG} = H(SK_{SG} || TID_S^{new} || TS_V)$ and sends an authentication acknowledgment message as $Msg_{D2G_3} = \langle SKV_{SG}, TS_V \rangle$ to the GWN via open channel.

- *Step D2G*$_4$: Once the GWN receives the acknowledgment message Msg_{D2G_3} from SN at TS_V^*, it validates the timestamp as $|TS_V^* - TS_V| \leq \Delta T$. If the condition is verified as valid, the GWN computes the session key verifier as $SKV_{GS} = H(SK_{GS} || TID_S^{new} || TS_V)$ and checks if the received SKV_{SG} matches with the computed SKV_{GS}. If it is valid, the GWN updates TID_S by TID_S^{new} in its secure database.

At the end of this phase, both SN and GWN share the same session key $SK_{GS} = SK_{SG}$. Finally, the summary of this phase is given in Figure 6.5.

IoT Smart Device (SN)	Gateway Node (GWN)
Pick $p_S \in Z_q^*$, timestamp TS_S	
Calculate $A_S = H(TID_S \|\| p_S \|\| TS_S) \cdot G$	
$x_S = H(pr_S \|\| p_S \|\| RID_S \|\| TS_S)$ $\oplus H(TC_S \|\| TID_S \|\| RID_S \|\| TS_S)$	Check if $\|TS_S^* - TS_S\| \leq \Delta T$? If so, check if $Sig_S \cdot G = A_S + H(x_S \|\| TID_S \|\| A_S \|\| TS_S \|\| TC_S) \cdot Pub_S$? If so,
$Sig_S = H(TID_S \|\| p_S \|\| TS_S) + H(x_S \|\| TID_S \|\| A_S \|\| TS_S \|\| TC_S) * pr_S \pmod q$	$y_S = H(pr_S \|\| p_S \|\| RID_S \|\| TS_S) = x_S \oplus H(TC_S \|\| TID_S \|\| RID_S \|\| TS_S)$
$Msg_{D2G_1} = \langle TID_S, A_S, x_S, Sig_S, TS_S \rangle \xrightarrow{}$ via public channel	generates $q_G \in Z_q^*$,timestamp TS_G
	$B_G = H(TID_G \|\| q_G \|\| RID_S \|\| TS_G) \cdot G$
	$DK_{GS} = H(TID_G \|\| q_G \|\| RID_S \|\| TS_G) \cdot A_S$
	$y_G = H(pr_G \|\| TID_G \|\| RID_G \|\| TC_S \|\| TS_G) \oplus H(TID_G \|\| TC_S \|\| TS_S \|\| TS_G \|\| RID_S)$
	$SK_{GS} = H(DK_{GS} \|\| y_S \|\| H(pr_G \|\| TID_G \|\| RID_G \|\| TC_S \|\| TS_G))$
checks if $\|TS_G^* - TS_G\| \leq \Delta T$?	generates $TID_S^{new} \in Z_q^*$
$DK_{SG} = H(TID_S \|\| p_S \|\| TS_S) \cdot B_G$	$TID_S^* = TID_S^{new} \oplus H(SK_{GS} \|\| TS_G)$
$z_G = H(pr_G \|\| TID_G \|\| RID_G \|\| TC_S \|\| TS_G) = y_G \oplus H(TID_G \|\| TC_S \|\| TS_S \|\| TS_G \|\| RID_S)$	$Sig_G = H(TID_G \|\| q_G \|\| RID_S \|\| TS_G) + H(TID_S \|\| TID_G \|\| TC_S \|\| y_G \|\| Pub_G \|\| TS_G) * pr_G \pmod q$
$SK_{SG} = H(DK_{SG} \|\| H(pr_S \|\| p_S \|\| RID_S \|\| TS_S) \|\| z_G)$	$Msg_{D2G_2} = \langle TID_G, B_G, y_G, Sig_G, TID_S^*, TS_G \rangle \xleftarrow{}$ via public channel
$Sig_G \cdot G = B_G + H(TID_S \|\| TID_G \|\| TC_S \|\| y_G \|\| Pub_G \|\| TS_G) \cdot Pub_G$	
$TID_S^{new} = TID_S^* \oplus H(SK_{GS} \|\| TS_G)$	
updates TID_S with TID_S^{new} in its memory	
generates timestamp TS_V	checks if $\|TS_V^* - TS_V\| \leq \Delta T$?
$SKV_{SG} = H(SK_{SG} \|\| TID_S^{new} \|\| TS_V)$	$SKV_{GS} = H(SK_{GS} \|\| TID_S^{new} \|\| TS_V)$
$Msg_{D2G_3} = \langle SKV_{SG}, TS_V \rangle \xrightarrow{}$ via open channel	checks if $SKV_{SG} \stackrel{?}{=} SKV_{GS}$
	GWN also updates TID_S by TID_S^{new} in its secure database
Both SN and GWN share the same secret key $SK_{SG} = SK_{GS}$	

Figure 6.5: Summary of D2G authentication phase

6.3.4 Dynamic smart node addition phase

This phase is executed in offline mode by the registration authority (RA) when an existing IoT smart device needs to be replaced due to its malfunction or suspicion of loss of security. To execute this phase, a new IoT smart device, say SN^{new} is deployed in a particular zone of the agriculture field after the following steps are performed:

- *Step* $DSNA_1$: The RA picks a real identity ID_S^{new}, temporary identity TID_S^{new}, and a random secret $s_1^{new} \in Z_q^*$. It then computes a pseudo-identity for SN^{new} as $RID_S^{new} =$

$H(ID_S^{new}||\ s_1^{new}||mk_{RA})$. Moreover, the RA picks a random private key $pr_S^{new} \in Z_q^*$, and computes the corresponding public key $Pub_S^{new} = pr_S^{new} \cdot G$ and a temporal credential for the SN^{new} as $TC_S^{new} = H(RID_S^{new}||\ pr_S^{new}||\ mk_{RA}||\ RTS_S^{new})$ where RTS_S^{new} is the current timestamp of registration of SN^{new}.

- *Step* $DSNA_2$: The RA then preloads SN^{new} with $\{(RID_S^{new}, TID_S^{new}, TC_S^{new}), H(\cdot), E_q(a,b), G, (pr_S^{new}, Pub_S^{new})\}$. In addition, the RA publishes Pub_S^{new} as SN^{new}'s public key.

6.3.5 Blockchain formation phase

In this phase, we consider the processes related to block creation, smart contract-based block validation and voting-based consensus for block addition.

1) Block creation

The blocks are generated by a gateway node (GWN) using the following steps:

- *Step* BC_1*:* On receiving the sensed data $SNDATA_i$ from a smart node SN_i, the GWN generates the "Elliptic Curve Digital Signature Algorithm (ECDSA)" signature on a transaction $Tx_i = (ID_G, TS_G, Z_i, SNDATA_i)$ using the private key pr_G of the GWN as $Sign_{Tx_i} = Sig_{pr_G}(H(ID_G\ ||\ TS_G\ ||\ Z_i\ ||\ SNDATA_i))$, where Z_i is the zone id to which SN_i belongs to, TS_G is the current timestamp and $Sig(\cdot)$ is the ECDSA signature generation algorithm [186]. The GWN then sends $(Tx_i, Sign_{Tx_i})$ to its associated edge server ES via public channel as this transaction is public. Similarly, other transactions belonging to the production process need to be private to a particular stakeholder while certain other data must be made available to some or all stakeholders to maintain transparency, and these transactions need to be encrypted using the public key of the GWN before generating signature $ESign_{Tx_i}$ on encrypted transactions $E_{Pub_G}(Tx_i)$ before sending them to the ES. Thus, some transactions are public and others are private in a block. As a result, a hybrid (consortium) blockchain has been considered.

- *Step* BC_2: The ES verifies each received transaction $(Tx_i, Sign_{Tx_i})$ or $(E_{Pub_G}(Tx_i), ESign_{Tx_i})$ by validating the $Sign_{Tx_i}$ or $ESign_{Tx_i}$ using the public key Pub_G of the GWN. If the signature is valid, the ES marks the (unencrypted) transaction Tx_i or encrypted transaction $E_{Pub_G}(Tx_i)$ as valid and proceeds for creating the partial block using n_t number of transactions. The ES then prepares a partial block $PBlock_i$ on the list

of n_t transactions, containing the block version *"Block_Ver"*, block creation timestamp *"TS_{ES}"*, Merkle Tree root *"MTR_{TX}"* based on the n_t unencrypted/encrypted transactions, owner of the block as an edge server (ES), public key of transactions verification (Pub_G), ECDSA signature $Sig_{ParBlock}$ on the list of n_t transactions, and partial block header using the private key pr_E of the edge server ES. The ES then sends the created partial block $\{Block\_Ver,\ MTR_{TX},\ TS_{ES},\ ES,\ Pub_G,\ Pub_E,\ (E_{Pub_G}(Tx_i, ESign_{Tx_i}), \cdots, (Tx_i, Sign_{Tx_i})),\ Sig_{ParBlock}\}$ to the cloud server CS via public channel, where the signature $Sig_{ParBlock}$ is generated on the partial block header and transactions using the private key pr_E of the ES.

- *Step* BC_3: The cloud server CS then validates the signature $Sig_{ParBlock}$ using the public key Pub_E of the ES that is present in the partial block. If the validation succeeds, the CS then adds the previous block hash ($Prev\_Block\_Hash$) and computes the current block hash Cur_Block_Hash on the entire block (partial block along with the partial block signature $Sig_{ParBlock}$, and completes $PBlock_i$ into a full block $Block_i$. Figure 6.6 depicts the skeleton of a full block.

2) Smart contract-based block validation

The expression "Code is Law" applies to smart contracts, where the technology enforces the rules and dictates what one can and cannot do. A smart contract is an application or a piece of code that gets executed on the blockchain automatically in a deterministic way. In Ethereum, the smart contracts are immutable, so once a smart contract is deployed, it cannot be modified. Also, the smart contracts in Ethereum are decentralized as there is no single entity that controls the smart contract state or execution, and all the nodes in the Ethereum network store the same contract in the same state. There are a few other general-purpose smart contract platforms besides Ethereum (viz., Cardano, Tron, Tezos, and Hyperledger Sawtooth). The proposed SCBAS-SF deploys a smart contract using "IF...THEN" semantics which allows a full block $Block_i$ to be added by a cloud server in the P2P cloud servers network. Algorithm 3 gives an overview of the smart contract processing as part of the block validation.

3) Voting-based consensus for block addition

The cloud servers (CS) in the blockchain network of the proposed SCBAS-SF employ a voting-based consensus algorithm to achieve consensus on a given block. The "Practical

Block Header	
Block Version ($Block\_Ver$)	Unique block version number
Previous Block Hash ($Prev\_Block\_Hash$)	Hash value of previous block
Merkle Tree Root (MTR_{TX})	Merkle tree root on transactions
Timestamp (TS_{ES})	Block creation time
Owner of Block	An edge server (ES)
Public key of transactions verification	Pub_G
Public key of block signer	Pub_E
Block Payload (Encrypted Transactions)	
Encrypted Transactions Tx_1	$(E_{Pub_G}(Tx_i), ESign_{Tx_i})$
	$\vdots$
Unencrypted Transactions Tx_2	$(Tx_i, Sign_{Tx_i})$
ECDSA signature on Partial Block	$Sig_{ParBlock}$
Current Block Hash (Cur_Block_Hash)	Hash value of current block

Figure 6.6: Structure of a full block $Block_i$ in the blockchain

Byzantine Fault Tolerance (PBFT)" has been applied in SCBAS-SF, which is one of the classical voting-based consensus protocols. In PBFT, one node acts as a *leader node* and other nodes are called *replicas*. The leader node is responsible for initiating a proposal to add a block to the blockchain, and the leader role gets changed from node to node in a round-robin fashion. PBFT declares consensus whenever it receives a positive response from $\frac{2}{3}$ of the nodes and allows at most $\frac{1}{3}$ of the nodes to be faulty or compromised in the system. Algorithm 3 provides a detailed workflow of the consensus process, including the smart contract validation.

Finally, the overall process flow of the proposed SCBAS-SF is depicted in Figure 6.7.

6.4 Security analysis

This section proves that the proposed scheme (SCBAS-SF) for smart farming stands strongly secure against potential attacks through the use of the widely recognized "Real-or-Random

Algorithm 3 Smart contract processing and consensus workflow of the blockchain

Input: $Block_i$: A full block having the structure as given in Figure 6.6 that is to be added to the blockchain, N: Total number of P2P nodes (cloud servers) in the blockchain network

Output: Block commit status (YES/NO)

Set $Magic\_Number = 2 * (N - 1)/3 + 1$
$CMP_i \leftarrow \Phi$ (empty)
Broadcast $Block_i$ to the replica nodes in the network to peers
for each replica cloud server node CS_j **do**
    /* Smart contract processing */
    Set $Consensus\_Vote_j = NO$
    Compute $Block\_Hash = H(Block_i)$
    if $(Block\_Hash = Curr\_Block\_Hash)$ **then**
        if (validation of $Sig_{ParBlock}$ using Pub_{ES} is successful) **then**
            Generate Merkle tree root (MTR'_{TX}) using the n_t transactions stored in the block payload
            if $(MTR'_{TX} = MTR_{TX})$ **then**
                Set $Consensus\_Vote_j = YES$
    Add $Consensus\_Vote_j$ to CMP_i
Set $App_{Count} \leftarrow 0$
for each vote V reply in CMP_i **do**
    if (V is YES) **then**
        Set $App_{Count} = App_{Count} + 1$
if $(App_{Count} \geq Magic\_Number)$ **then**
    Add block $Block_i$ into the blockchain
    Broadcast block commit status as YES to the blockchain network

(ROR) model" [43] in formal security analysis, a non-mathematical (informal) security analysis and also formal security verification under the widely recognized "Automated Validation of Internet Security Protocols and Applications (AVISPA)" software validation tool [9].

6.4.1 Formal security using ROR model

In the proposed SCBAS-SF, one session key is established between two IoT smart devices during the D2D authentication phase, and another session key is also established between an

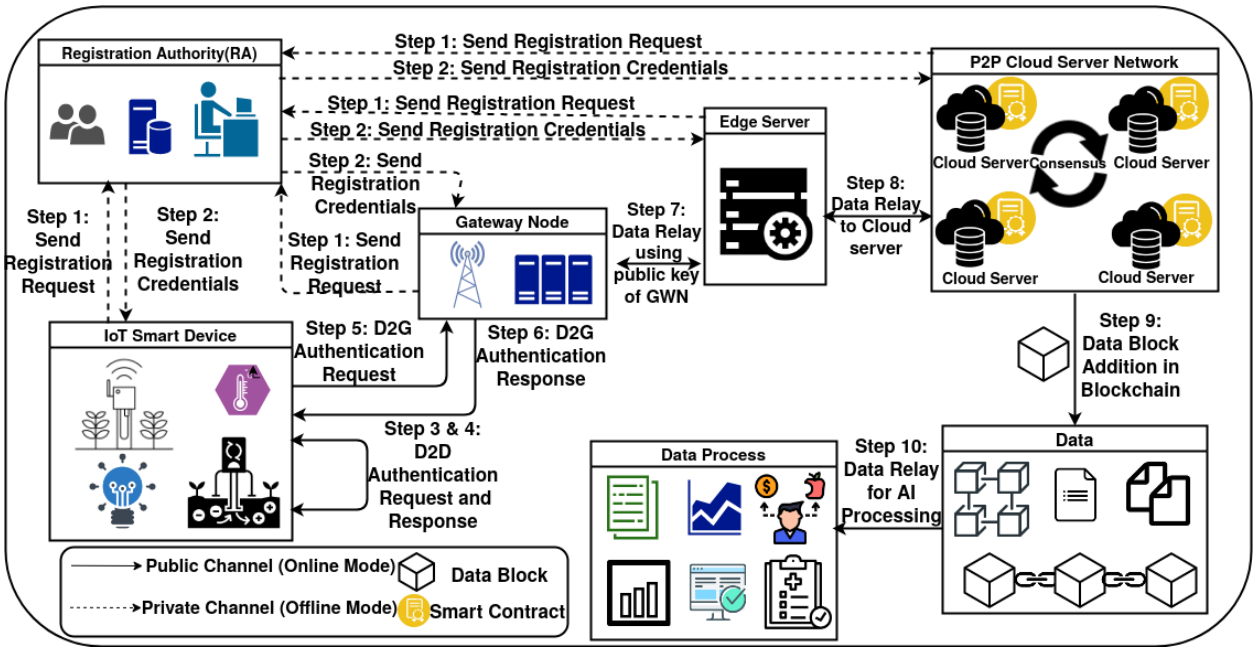

Figure 6.7: Overall process flow of the proposed scheme

IoT smart device SN and the gateway node GWN during the D2G authentication phase. In this section, we formulate that the created session keys are secure under the ROR oracle model [43]. The security of the session key has been proved in Theorem 6.1 based on the definition of semantic security as given in Definition 6.1. All the involved entities have access to a "one-way cryptographic hash function" $H(\cdot)$ that is accounted as a random oracle, say $Hash$. In addition, the queries shown in Table 6.2 are available to an adversary $\mathcal{A}$.

Two IoT smart devices entities SN_1 and SN_2 participate in the D2D authentication phase to establish a session key $SK_{S1S2} = (SK_{S2S1})$ (see Section 1), and the entities GWN and its associated IoT smart device (SN) generate a session key $SK_{SG} = (SK_{GS})$ in the D2G authentication phase (see Section 2). Let $\Psi_{SN_1}^{l_1}$, $\Psi_{SN_2}^{l_1}$ and $\Psi_{GWN}^{l_2}$ denote the l_1^{th}, l_2^{th} and l_3^{th} instances of SN_1, SN_2 and GWN, respectively, which are known as "random oracles".

Definition 6.1 (Semantic security). *Let $Adv_{\mathcal{A}}^{SCBAS-SF}(t_p)$ denote the "advantage of an adversary $\mathcal{A}$, running in polynomial time t_p in breaking the semantic security of the proposed SCBAS-SF in order to derive the session key $SK_{S1S2} = SK_{S2S1}$ between two IoT smart devices SN_1 and SN_2 in the D2D authentication phase and the session key $SK_{SG} = (SK_{GS})$ between a smart device SN and its gateway node GWN in the D2G authentication phase of the proposed SCBAS-SF in a particular session". Then, $Adv_{\mathcal{A}}^{SCBAS-SF}(t_p) = |2Pr[b' = b] - 1|$, where b and b' are the "correct" and "guessed" bits, respectively.*

Table 6.2: Queries and their purposes

Query	Purpose
$Execute(\Psi^{l_1}_{SN_1}, \Psi^{l_2}_{SN_2}, \Psi^{l_3}_{GWN})$	$\mathcal{A}$ eavesdrops on the communicated messages between SN_1, SN_2 and GWN with this query
$CorruptSD(\Psi^{l}_{SN_i})$	$\mathcal{A}$ can procure all the pre-stored secret credentials of any compromised smart device SN with this query
$Reveal(\Psi^{l})$	$\mathcal{A}$ uses this query to procure the session key shared between Ψ^l and its respective participant
$Test(\Psi^{l})$	$\mathcal{A}$ verifies if the session key between SN_1 and SN_2, and also between SN and GWN are original or random using this query

Theorem 6.1. *Assume that an adversary $\mathcal{A}$ executes in polynomial time t_p and attempts to obtain the session key SK_{S1S2} $(= SK_{S2S1})$ established between two smart devices SN_1 and SN_2 during the D2D authentication phase, and the session key SK_{SG} $(= SK_{GS})$ established a smart device SN and its gateway node GWN for a particular session during the D2G authentication phase of the proposed SCBAS-SF. If q_h, $|Hash|$ and $Adv^{ECDDHP}_{\mathcal{A}}(t_p)$ represent the "number of $Hash$ queries", the "range space of a one-way collision-resistant hash function $H(\cdot)$" and the "advantage in breaking the Elliptic Curve Decisional Diffie-Hellman Problem (ECDDHP)", respectively, then*

$$Adv^{SCBAS-SF}_{\mathcal{A}}(t_p) \leq \frac{q_h^2}{|Hash|} + 2Adv^{ECDDHP}_{\mathcal{A}}(t_p).$$

Proof. The proof of this theorem is followed in a similar way that was done in [73, 74, 227]. In SCBAS-SF, a series of three games are designed to be executed by the adversary $\mathcal{A}$, say $Game^{\mathcal{A}}_l$, $(l = 0, 1, 2)$, where $Success^{\mathcal{A}}_{Game_l}$ is an event such that $\mathcal{A}$ can guess a random bit b in $Game^{\mathcal{A}}_l$ correctly. Thus, the advantage of $\mathcal{A}$ is the success probability to win $Game^{\mathcal{A}}_l$ given as $Adv^{SCBAS-SF}_{\mathcal{A},Game_l} = Pr[Success^{\mathcal{A}}_{Game_l}]$. The games played by the adversary $\mathcal{A}$ against the proposed scheme $SCBAS-SF$ are as follows:

- $Game^{\mathcal{A}}_0$: In this game, $\mathcal{A}$ launches an actual attack against SCBAS-SF by picking a random bit b before the start of the game $Game^{\mathcal{A}}_0$. By the "semantic security as defined in Definition 6.1", it follows that

$$Adv^{SCBAS-SF}_{\mathcal{A}}(t_p) = |2Adv^{SCBAS-SF}_{\mathcal{A},Game_0} - 1|. \tag{6.1}$$

- $Game_1^{\mathcal{A}}$: In this game, $\mathcal{A}$ launches an eavesdropping attack by running the *Execute* query followed by the *Test* query. The outcome of the *Test* query helps $\mathcal{A}$ to decide whether the values extracted from the *Reveal* query are the original session keys or some random keys. With the *Execute* query, $\mathcal{A}$ intercepts the messages Msg_{D2D_1}, Msg_{D2D_2} and Msg_{D2D_3} during D2D authentication phase and Msg_{D2G_1}, Msg_{D2G_2} and Msg_{D2G_3} during D2G authentication phase. The session key between two smart devices SN_1 and SN_2 is $SK_{S1S2} = x_{S1} \cdot Y_{S2} = SK_{S2S1} = y_{S2} \cdot X_{S1}$ where $Y_{S2} = y_{S2} \cdot G$ and $y_{S2} = H(TID_{S2}|| \ RID_{S2}|| \ TC_{S2}|| \ r_{S2}|| \ pr_{S2}|| \ TS_{S2})$, $X_{S1} = x_{S1} \cdot G$ and $x_{S1} = H(RID_{S1}|| \ TID_{S1}|| \ TC_{S1}|| \ pr_{S1}|| \ TS_{S1}|| \ r_{S1})$. This session key depends on both the temporal (short-term) secrets (r_{S1}, r_{S2}) and long-term secrets (RID_{S1}, RID_{S2}, TC_{S1}, TC_{S2}, pr_{S1}, pr_{S2}). Moreover, the session key $SK_{GS} = H(DK_{GS}|| \ y_S|| \ H(pr_G|| \ TID_G|| \ RID_G|| \ TC_S|| \ TS_G))$ where $DK_{GS} = H(TID_G|| \ q_G|| \ RID_S|| \ TS_G) \cdot A_S$ also depends on temporal (short-term) secrets (p_S, q_G, x_S, y_G) and long-term secrets (RID_S, RID_G, TC_S, pr_S, pr_G). To add to the security, "collision-resistant one-way hash function $h(\cdot)$" is used to protect the secret parameters. This suggests that the success probability of $\mathcal{A}$ cannot be increased by simple capturing of messages to reveal the session keys SK_{S1S2} ($= SK_{S2S1}$) and SK_{SG} ($= SK_{GS}$). Therefore, $Game_0^{\mathcal{A}}$ and $Game_1^{\mathcal{A}}$ are indistinguishable under the eavesdropping attack. Thus, it follows that

$$Adv_{\mathcal{A},Game_1}^{SCBAS-SF} = Adv_{\mathcal{A},Game_0}^{SCBAS-SF}. \tag{6.2}$$

- $Game_2^{\mathcal{A}}$: Under this game, $\mathcal{A}$ launches an active attack by simulating $Hash$ queries and execution of computational ECDDHP problem. In the D2D authentication phase, the session key is derived as $SK_{S2S1} = y_{S2} \cdot X_{S1}$ by SN_1 and $SK_{S1S2} = x_{S1} \cdot Y_{S2}$ by SN_2. $\mathcal{A}$ can obtain X_{S1} and Y_{S2} from Msg_{D2D_1} and Msg_{D2D_2} in transit. For $\mathcal{A}$ to derive the session key from known X_{S1} and Y_{S2}, it needs to first solve the computational ECDDHP to compute the private x_{S1} and y_{S2} which are based on the secrets (r_{S1}, r_{S2}) unknown to $\mathcal{A}$. Then, it needs to simulate $Hash$ queries to compute x_{S1} and y_{S2} since the private x_{S1} and y_{S2} are also protected by the "collision-resistant one-way hash function $H(\cdot)$". In the D2G authentication phase, the session key is derived by SN as $SK_{SG} = H(DK_{SG}|| \ H(pr_S|| \ p_S|| \ RID_S|| \ TS_S)|| \ Z_G) = SK_{GS}$. This session key consists of three parts: a) the Diffie-Hellman type key $DK_{GS} = (DK_{SG})$, b) private credentials of SN as $H(pr_S|| \ p_S|| \ RID_S|| \ TS_S)$ and c) private credentials of GWN as $H(pr_G|| \ TID_G|| \ RID_G|| \ TC_S|| \ TS_G)$. $\mathcal{A}$ can obtain the following from the messages in transit: A_S and B_G needed in DK_{GS} and DK_{SG}, x_S that hides $H(pr_S|| \ p_S|| \ RID_S||$

TS_S) to be extracted as y_S by the GWN, y_G that hides $H(pr_G||\ TID_G||\ RID_G||\ TC_S||\ TS_G)$ to be extracted as z_G by SN. $\mathcal{A}$ needs to execute the $Hash$ queries to obtain y_S, z_G, $A_S = H(TID_S||\ p_S||\ TS_S) \cdot G$ and $B_G = H(TID_G||\ q_G||\ RID_S||\ TS_G) \cdot G$ which are based on the secrets p_S and q_G unknown to $\mathcal{A}$ and protected by the "collision-resistant $H(\cdot)$". Then, it needs to solve the computational ECDDHP to extract $H(TID_S||\ p_S||\ TS_S)$ and $H(TID_G||\ q_G||\ RID_S||\ TS_G)$ from known A_S, B_G and G. Thus, it can be understood that the session keys, during both the D2D and D2G authentication phases, can only be derived if $\mathcal{A}$ can solve both $Hash$ queries and ECDDHP. The games $Game_1^{\mathcal{A}}$ and $Game_2^{\mathcal{A}}$ become indistinguishable if the $Hash$ queries simulation and the computational ECDDHP are excluded in game $Game_2^{\mathcal{A}}$. The advantage of solving ECDDHP and the birthday paradox for finding the hash collision produce the following relationship:

$$|Adv_{\mathcal{A},Game_1}^{SCBAS-SF} - Adv_{\mathcal{A},Game_2}^{SCBAS-SF}| \leq \frac{q_h^2}{2|Hash|} + Adv_{\mathcal{A}}^{ECDDHP}(t_p) \tag{6.3}$$

$\mathcal{A}$ has executed all the queries earlier except guessing a bit to win the game $Game_2^{\mathcal{A}}$, leading to $Adv_{\mathcal{A},Game_2}^{SCBAS-SF} = \frac{1}{2}$.

Eqs. (6.1), (6.2) and (6.3), and the use of triangular inequality produce the following derivation:

$$\begin{aligned} \frac{1}{2}.Adv_{\mathcal{A}}^{SCBAS-SF}(t_p) &= |Adv_{\mathcal{A},Game_0}^{SCBAS-SF} - \frac{1}{2}| \\ &= |Adv_{\mathcal{A},Game_1}^{SCBAS-SF} - Adv_{\mathcal{A},Game_2}^{SCBAS-SF}| \\ &\leq \frac{q_h^2}{2|Hash|} + Adv_{\mathcal{A}}^{ECDDHP}(t_p). \end{aligned} \tag{6.4}$$

Finally, if we multiply both sides of Eq. (6.4) by "a factor of 2", we arrive to the final result: $Adv_{\mathcal{A}}^{SCBAS-SF}(t_p) \leq \frac{q_h^2}{|Hash|} + 2Adv_{\mathcal{A}}^{ECDDHP}(t_p)$. □

6.4.2 Informal security analysis

Through the non-mathematical security analysis, we show that the proposed scheme (SCBAS-SF) resists several known attacks.

1) **Ephemeral Secret Leakage (ESL) attack**: During the D2D authentication phase, an IoT smart device SN_1 computes the shared session key with its neighbor IoT smart device SN_2 as $SK_{S1S2} = x_{S1} \cdot Y_{S2}$ where $Y_{S2} = y_{S2} \cdot G$ and $y_{S2} = H(TID_{S2}||\ RID_{S2}||$

$TC_{S2}||\ r_{S2}||\ pr_{S2}||\ TS_{S2})$. SN_2 also computes the shared session key shared with SN_1 as $SK_{S2S1} = y_{S2} \cdot X_{S1} = SK_{S1S2}$ where $X_{S1} = x_{S1} \cdot G$ and $x_{S1} = H(RID_{S1}||\ TID_{S1}||\ TC_{S1}||\ pr_{S1}||\ TS_{S1}||\ r_{S1})$. The computed session key depends on both the temporal (short-term) secrets (r_{S1}, r_{S2}) and long-term secrets (RID_{S1}, RID_{S2}, TC_{S1}, TC_{S2}, pr_{S1}, pr_{S2}).

During the D2G authentication phase, the gateway GWN computes the shared session key with the accessed IoT smart device SN as $SK_{GS} = H(DK_{GS}||\ y_S||\ H(pr_G||\ TID_G||\ RID_G||\ TC_S||\ TS_S))$ where $DK_{GS} = H(TID_G||\ q_G||\ RID_S||\ TS_G) \cdot A_S$. On the other side, SN also computes that shared session key with GWN as $SK_{SG} = H(DK_{SG}||\ H(pr_S||\ p_S||\ RID_S||\ TS_S)||\ Z_G) = SK_{GS}$ where $DK_{SG} = H(TID_S||\ p_S||\ TS_S) \cdot B_G$. The computed session key also depends on both the temporal (short-term) secrets (p_S, q_G, x_S, y_G) and long-term secrets (RID_S, RID_G, TC_S, pr_S, pr_G).

We now consider the following two cases:

- **Case 1.** In D2D authentication phase, if only the temporal (short-term) secrets (r_{S1}, r_{S2}) and long-term secrets (RID_{S1}, RID_{S2}, TC_{S1}, TC_{S2}, pr_{S1}, pr_{S2}). Similarly, in D2G authentication phase, if (p_S, q_G, x_S, y_G) are compromised, the session key SK_{GS} ($= SK_{SG}$) cannot be compromised without possessing the long-term secrets (RID_S, RID_G, TC_S, pr_S, pr_G).
- **Case 2.** If only the long-term secrets (RID_{S1}, RID_{S2}, TC_{S1}, TC_{S2}, pr_{S1}, pr_{S2}) in D2D authentication phase and (RID_S, RID_G, TC_S, pr_S, pr_G) in D2G authentication phase are compromised, the session keys SK_{S1S2} ($= SK_{S2S1}$) and SK_{SG} ($= SK_{GS}$) cannot be derived without possessing the short-term secrets (r_{S1}, r_{S2}) and (p_S, q_G, x_S, y_G).

Thus, the session is compromised when both the short and long-term secrets are compromised by an adversary. Based on the CK-adversary model, the proposed SCBAS-SF is then resilient against ESL attacks.

2) **Privileged insider attack**: The registration of the IoT smart devices, the gateway nodes, the edge servers, and the cloud servers is performed by the registration authority (RA) by pre-loading secret credentials into the entities and does not require the passing of any secret credentials via the public channel. Thus, a privileged insider will not possess any secret credentials to attack the system; hence, the proposed SCBAS-SF is resilient against "privileged insider attack".

3) **Replay attack**: Assume that an adversary, say $\mathcal{A}$ intercepts the messages Msg_{D2D_1}, Msg_{D2D_2} and Msg_{D2D_3} during D2D authentication phase, and Msg_{D2G_1}, Msg_{D2G_2} and Msg_{D2G_3} during D2G authentication phase of the proposed SCBAS-SF. It can be observed that every message includes either timestamps or random secrets, or both. Every receiver will validate these values before any processing. If the timestamps are not validated, the receiver will discard the received message without any processing. This prevents $\mathcal{A}$ from replaying previous messages, and thus, SCBAS-SF is resilient against replay attacks.

4) **Man-in-the-middle attack**: Assume that an adversary $\mathcal{A}$ intercepts the messages Msg_{D2D_1}, Msg_{D2D_2}, Msg_{D2D_3} in D2D authentication phase and Msg_{D2G_1}, Msg_{D2G_2}, Msg_{D2G_3} in D2G authentication phase, and tries to forward them to the intended recipients after tampering the message contents. In D2D authentication phase, $Sign_{S_1}$ uses x_{S_1} that is based on the secret credential r_{S1} private to SN_1, and hence, it cannot be tampered with. Similarly, $Sign_{S_2}$ uses y_{S_2} that is based on the secret credential r_{S2} private to SN_2. In D2G authentication phase, Sig_S uses the secret p_S and Sig_G uses the secret q_G. If the signatures are tampered, it can be immediately identified at the receiver by verifying the signature. Thus, the proposed SCBAS-SF is resilient against "man-in-the-middle attack".

5) **Impersonation attacks**: Suppose an adversary $\mathcal{A}$ intercepts the messages Msg_{D2D_1}, Msg_{D2D_2} and Msg_{D2D_3} during D2D authentication phase and Msg_{D2G_1}, Msg_{D2G_2} and Msg_{D2G_3} during D2G authentication phase of the proposed scheme. We consider the following two cases:

   - *Gateway node impersonation attack*: In this attack, during the D2G authentication phase, assume that $\mathcal{A}$ tries to deceive the smart device SN by impersonating the gateway node GWN and fabricating a message $Msg^a_{D2G_2} = \langle TID^a_G, B^a_G, y^a_G, Sig^a_G, TID^{a*}_S, TS^a_G \rangle$. The adversary $\mathcal{A}$ needs to compute $B_G = H(TID_G || q_G || RID_S || TS_G) \cdot G$ which requires to generate timestamp TS^a_G and random secret $q^a_G \in Z^*_q$ to compute y^a_G and also other long-term secrets RID_G, RID_S and TC_S. Hence, the proposed scheme is resilient against gateway node impersonation attacks.
   - *IoT smart device impersonation attack*: To impersonate the smart device SN in the D2G authentication phase, the adversary $\mathcal{A}$ needs to fabricate the valid messages, say $Msg^a_{D2D_1} = \langle TID^a_{S1}, X^a_{S1}, Sig^a_{S1}, Pub^a_{S1}, TS^a_{S1} \rangle$ and $Msg^a_{D2D_2} = \langle TID^a_{S2}, Y^a_{S2}, Sig^a_{S2}, Pub^a_{S2}, TS^a_{S2} \rangle$. $\mathcal{A}$ generates timestamps TS^a_{S1} and TS^a_{S2}, and random secrets r^a_{S1} and r^a_{S2}. However, $\mathcal{A}$ does not have access to the secret credentials RID_{S1},

TC_{S1},RID_{S2} and TC_{S2}. Hence, $\mathcal{A}$ cannot send fabricated messages on behalf of SN_1 or SN_2. In a similar way, $\mathcal{A}$ can not send the fabricated message $Msg^a_{D2G_1}$ on behalf of the GWN. Since the secrets are unknown to $\mathcal{A}$, IoT smart device impersonation attack is protected against in the proposed SCBAS-SF.

6) **Physical smart device capture attacks**: If an adversary $\mathcal{A}$ physically seizes an IoT smart device SN due to the unguarded environment of IoT-enabled intelligent precision agriculture as discussed in the threat model in Section 6.1.2, he/she can extract all its credentials using the "power analysis attacks" [200]. This risks exposure of the information $\{(RID_S, TID_S, TC_S), H(\cdot), E_q(a,b), G, (pr_S, Pub_S)\}$ stored in SN along with its sensed data. Even if the extracted credentials are leaked, these cannot hamper the security of communication with other non-compromised smart devices SN and gateway node GWN as credentials are unique to the compromised device along with the session keys established. In a similar way, other non-compromised smart devices can also communicate securely among themselves even if the credentials from SN's memory are known to the adversary $\mathcal{A}$. Hence, SCBAS-SF is secure against "physical smart device capture attack".

7) **Denial-of-Service (DoS) attacks**: The usage of current timestamps in every exchanged message ensures that multiple messages from an adversary are easily detected by checking the timestamp at the receiver end (as discussed in Section 3), and the messages are not processed further. Therefore, the resources used by the entities cannot be consumed by the adversary as the computation is based on lightweight cryptographic operations, such as hash and ECC point addition/multiplication computations. As a result, SCBAS-SF is resilient against DoS attacks.

8) **Anonymity and untraceability**: The messages communicated between two IoT smart devices SN_1 and SN_2, and between the gateway node GWN and its IoT smart device SN during D2D and D2G authentication phases are Msg_{D2D_1}, Msg_{D2D_2} and Msg_{D2D_3}, and Msg_{D2G_1}, Msg_{D2G_2} and Msg_{D2G_3}, respectively. All the messages use only temporal identities and not the real/pseudo-identities of SN and GWN. In addition, the messages are all distinct due to unique and random components. Thus, it is not possible for an adversary to identify or trace the entities involved in communication over successive sessions. Therefore, both anonymity and untraceability properties are preserved in SCBAS-SF.

6.4.3 Formal security verification using AVISPA tool

In the proposed scheme (SCBAS-SF), the registration and D2D authentication phases are implemented with the three basic roles for the registration authority (RA), two IoT smart devices SN_1 and SN_2, and also the registration and D2G authentication phase are implemented using the basic roles for the registration authority (RA), an IoT smart device (SN) and a gateway node (GWN) along with the two mandatory roles for the session and environment. After that, we have simulated the proposed scheme for D2D and D2G authentication phases using the broadly accepted "SPAN, the Security Protocol ANimator for AVISPA" tool as described in Appendix A.

It is worth noticing that SCBAS-SF makes utilization of the bitwise XOR operation. Currently, out of four backends, two backends: SATMC and TA4SP, do not support bitwise XOR operation. Due to this reason, the simulation results under the SATMC and TA4SP backends will come as "inconclusive," and we have omitted such simulation results in this chapter. Under OFMC and CL-AtSe backends, the simulation results of the D2D authentication phase are shown in Figure 6.8, whereas the simulation results for the D2G authentication phase are also shown in Figure 6.9. The simulation results clearly demonstrate that SCBAS-SF is secure against both "replay" and "man-in-the-middle" attacks in both cases.

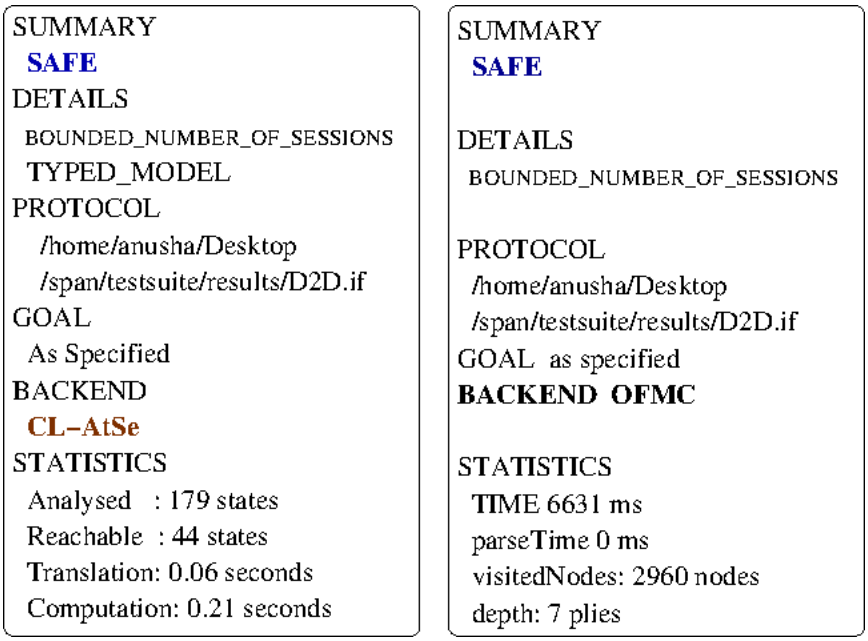

Figure 6.8: AVISPA simulation results for D2D authentication

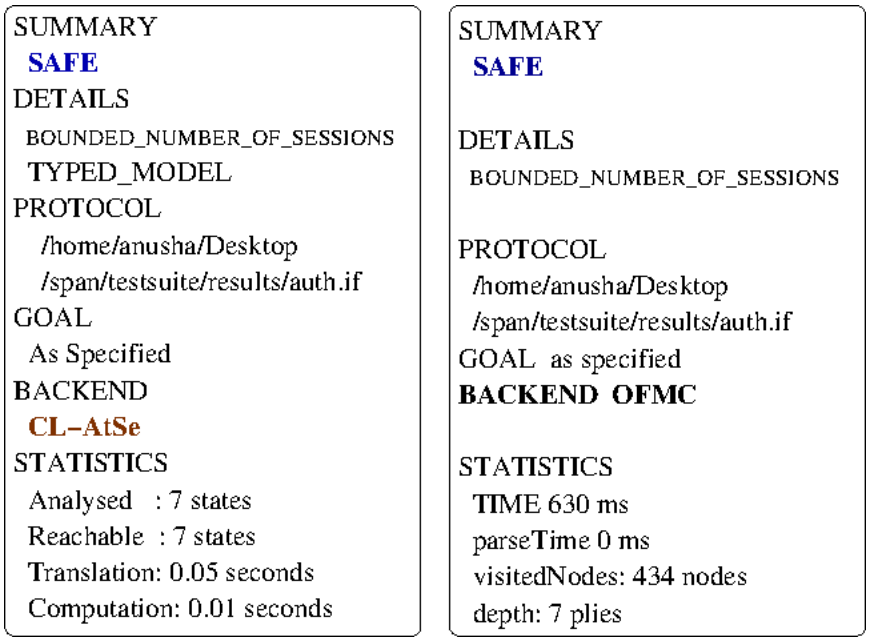

Figure 6.9: AVISPA simulation results for D2G authentication

6.5 Blockchain-based implementation

We have simulated the proposed SCBAS-SF on the Hyperledger Sawtooth [172] blockchain platform using the "Practical Byzantine Fault Tolerance (PBFT)" consensus algorithm [95]. Hyperledger Sawtooth offers a flexible and modular architecture that separates the core system from the application domain, and smart contracts can specify the business rules for applications without needing to know the underlying design of the core system. The nodes participating in the consensus process of a Sawtooth chain are called "validator" nodes. The Sawtooth framework facilitates modeling smart contracts as a state machine, often called a "transaction processor". A transaction processor is a pluggable module that gets registered with the validator node in a Sawtooth chain. After passing through the distributed log, transactions are routed to the appropriate transaction processor by each validator node. The sawtooth platform is agnostic to the transaction processing language and supports various high-level programming languages, such as Java and Python.

In the proposed SCBAS-SF, we have considered that the $Block\_Ver$, TS_{ES}, MTR_{TX}, Pub_E, Pub_G, $Sign_{TX}$, $ESign_{TX}$, Cur_Hash_Block, $Prev\_Block\_Hash$ and ECDSA signature are of sizes 32, 32, 256, 320, 320, 320, 320, 256, 256 and 320 bits, respectively, where the hash output is taken as 256 bits by utilizing the SHA-256 hashing algorithm. Also, the values (ID_G, TS_G, Z_i, $SNDATA_i$) which are stored in a transaction are of 160, 32, 32 and

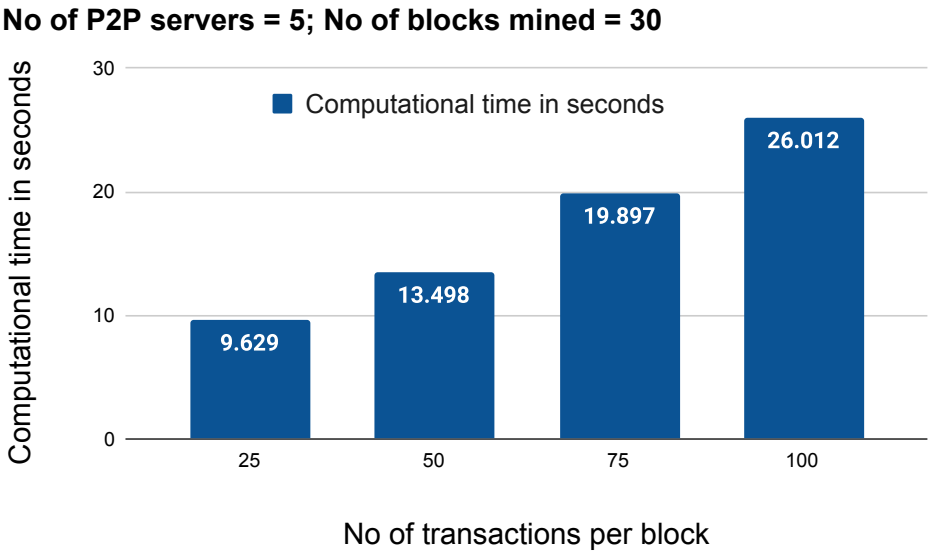

Figure 6.10: Blockchain simulation results for Case I

1024 bits, respectively. For simplicity, it is assumed that there are $n_t/2$ transactions are encrypted and the remaining $n_t/2$ transactions are unencrypted. Thus, a full block $Block_i$ shown in Figure 6.6 requires $1952 + 1264n_t$ bits, where an encrypted transaction $Tx_i = (ID_G, TS_G, Z_i, SNDATA_i)$ needs 1264 bits assuming the sensing data $SNDATA_i$ is of 1024 bits.

In our simulation, each validator node in the Sawtooth network has the configuration "Ubuntu 18.04, Intel(R) Core(TM) i9-9880H CPU @ 2.30 GHz, 2GB RAM". We have then evaluated the performance of the SCBAS-SF in the following two scenarios.

- **Case I**: The simulation is performed with a varied number of blocks with a fixed number of transactions (60) per block. Figure 6.10 shows the computational time in milliseconds versus the number of blocks mined.

- **Case II**: The simulation is performed with a fixed number of blocks (30) with a varied number of transactions per block. Figure 6.11 shows the computational time in milliseconds versus the number of transactions per mined block.

In both cases, it is worth observing that the computational time linearly increases when the number of blocks and the number of transactions per block are increased.

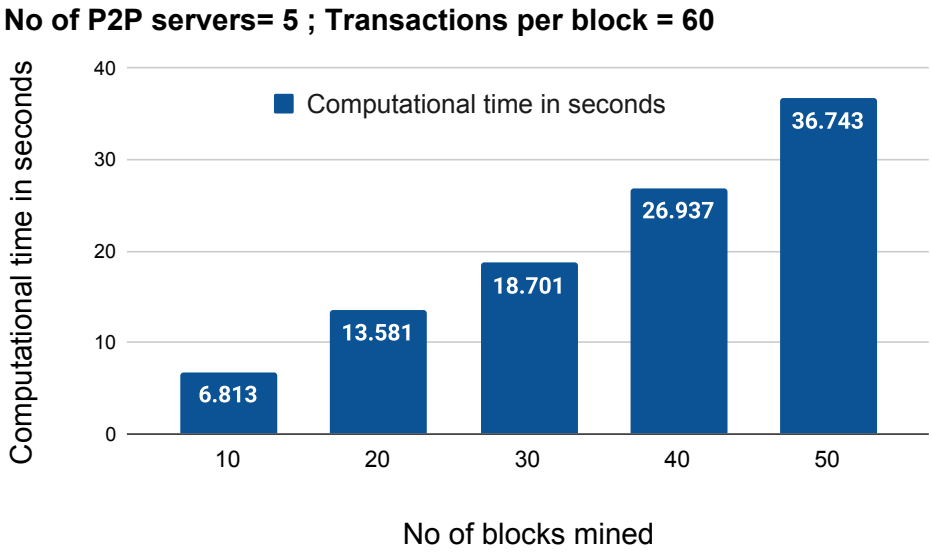

Figure 6.11: Blockchain simulation results for Case II

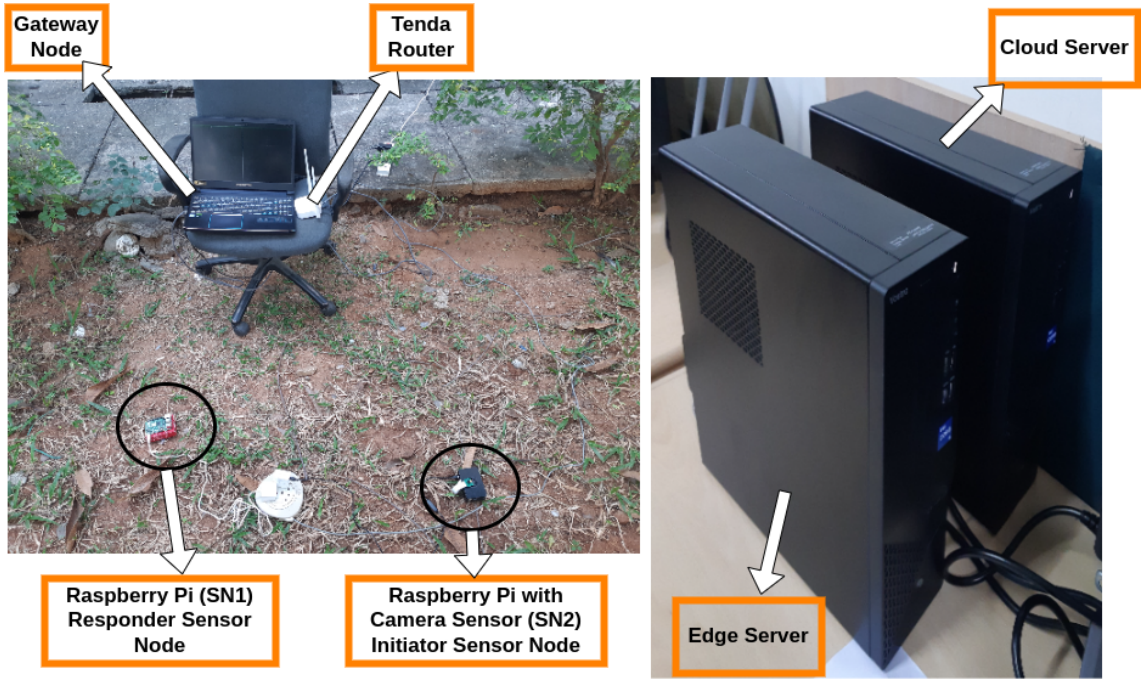

Figure 6.12: Testbed setup for real-time implementation of SCBAS-SF

Table 6.3: Summary of hardware configurations for real-time testbed

Entity	Device	Specifications
Camera sensor	Raspberry PI Camera Rev 1.3 P5V04A	Pixel Count: 2592 x 1944(5-megapixel) Lens: f=3.57 mm, f/2.8 View Angle: 65 degrees Focusing Range: 0.69m to infinity at 1.38m Support: 1080p@30fps with codec H.264 (AVC), 720p@60fps and 640x480p@60/90 fps video record Interface: CSI, 15cm flex cable
IoT smart devices (SN_1, SN_2)	Raspberry Pi 4 Model B 2BG RAM	Quad-Core 64-bit Broadcom BCM2711, Cortex A72 (ARM v8) Processor SoC @ 1.5GHz, 2.4 GHz and 5.0 GHz IEEE 802.11b/g/n/ac wireless LAN Bluetooth 5.0, BLE Gigabit Ethernet 2 × USB 3.0 ports 2 × USB 2.0 ports Memory: 2GB LPDDR4
Gateway node (GWN)	Laptop	Intel core i7 processor, 16 GB RAM and 256 GB SSD with Ubuntu Desktop 22.04 LTS operating system
Edge server (ES)	Desktop CPUs	Intel core 12th gen processor, 16 GB RAM and 500 GB SSD
Cloud server (CS)	Desktop CPUs	Intel core 12th gen processor, 16 GB RAM and 500 GB SSD

6.6 Real-time testbed implementation

The proposed scheme SCBAS-SF has been implemented using the proposed network model on a real-time testbed setup. This section details the configurations of the hardware and networking devices used for the testbed setup followed by a detailed explanation of the execution process. The final obtained outputs have also been presented and explained in detail. Figure 6.12 shows the testbed setup with all the devices configured for execution.

Table 6.4: Summary of network configurations for real-time testbed

Entity	Device	Specifications
Router	Tenda N300	Model N301, IP address: 192.168.0.1, Power: 9V 600mA 300 Mbps Speed, LAN/WAN 10/100 Frequency: 2.4 GHz, Single frequency band, External antenna, Number of antennae=2, Number of USB Ports=0, Number of LAN ports = 3, Number of WAN ports=1, Input type RJ-45 (Ethernet Cable) 2.4 GHz IEEE 802.11b/g/n, wireless LAN, 3G connectivity Encryption: WPA2-PSK, 64/128-bit WEP, WPS, WPA-PSK VPN: VPN Pass-through (PPTP / L2TP), Virtual Server: Port Forwarding, DMZ Host

6.6.1 Hardware configurations

Thr proposed model requires two IoT smart devices for SN_1 and SN_2, a computer for the gateway node GWN and two servers for the edge server ES and the cloud server CS. Table 6.3 gives a summary of the hardware devices and their configurations used for the testbed setup in the real-time implementation of the proposed model.

6.6.2 Network configurations

The IoT smart device SN_1 and SN_2 are connected via wireless local area network (LAN) using the router. The gateway node is connected to the same network via a category 5 (CAT5) Ethernet cable. The edge server ES and cloud server CS and the gateway node GWN are connected via the Internet. Table 6.4 gives a summary of the network devices and their configurations used for the testbed setup in the real-time implementation of the proposed model.

6.6.3 Execution results and discussion

We have used a camera sensor to capture an image of the farming field and sent the captured image from the IoT smart device SN_2 to the the cloud server CS via the IoT smart device SN_2, the gateway node GWN and the edge server ES. Multiple python scripts are run for the registration, authentication and encrypted data exchange phases. The code can be found

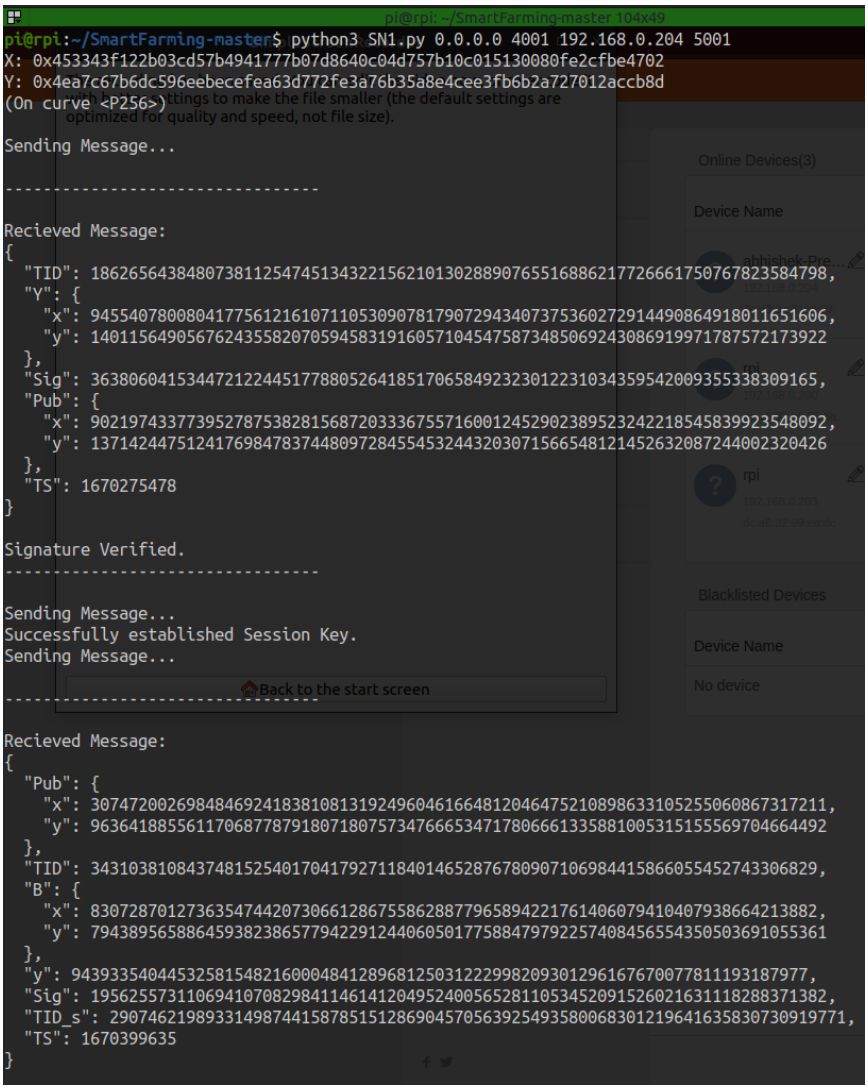

Figure 6.13: Execution and output terminal of IoT smart devices SN1 in D2D phase

at the github link: https://github.com/RajWorking/SmartFarming. Figure 6.13 shows the output terminal of the execution of registration and authentication of the two IoT smart devices SN_1 and SN_2 in the $D2G$ phase. The device SN_1 acts as the responder and is executed first to be in listening mode. The device SN_2 acts as the initiator node. It captures an image of the farm using its camera sensor, performs registration phase, executes the $D2D$ phase with SN_1, and sends the captured image encrypted with the established session key to the SN_1. The output terminal also shows the SN_2 authenticating with the gateway node GWN. Figure 6.15 shows SN_1 sending the image to GWN.

Figure 6.16 shows the authentication and session key establishment in $D2G$ phase at the

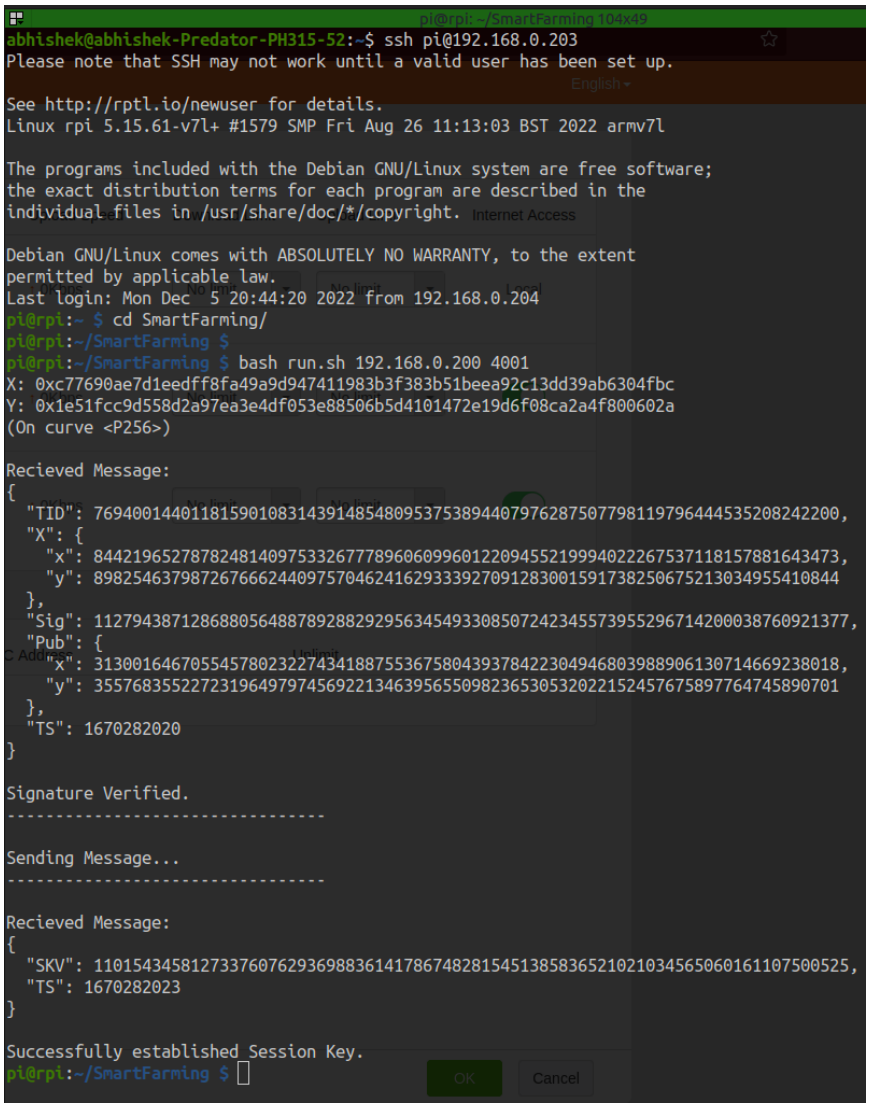

Figure 6.14: Execution and output terminal of IoT smart device SN2 in D2D phase

gateway node. Once the key is established, the image file encrypted with the established session key between SN_1 and GWN is received at GWN. GWN then decrypts the file with the same session key, encrypts with the public key of ES and sends to the edge server ES.

Figure 6.17 shows the ES receiving the file, decrypting it, encrypting with the public key of CS and sending the encrypted file to CS. Figure 6.18 shows the CS receiving the file and storing it. The image file captured by SN_2 and stored at the CS as `img.jpg` is shown in Figure 6.19.

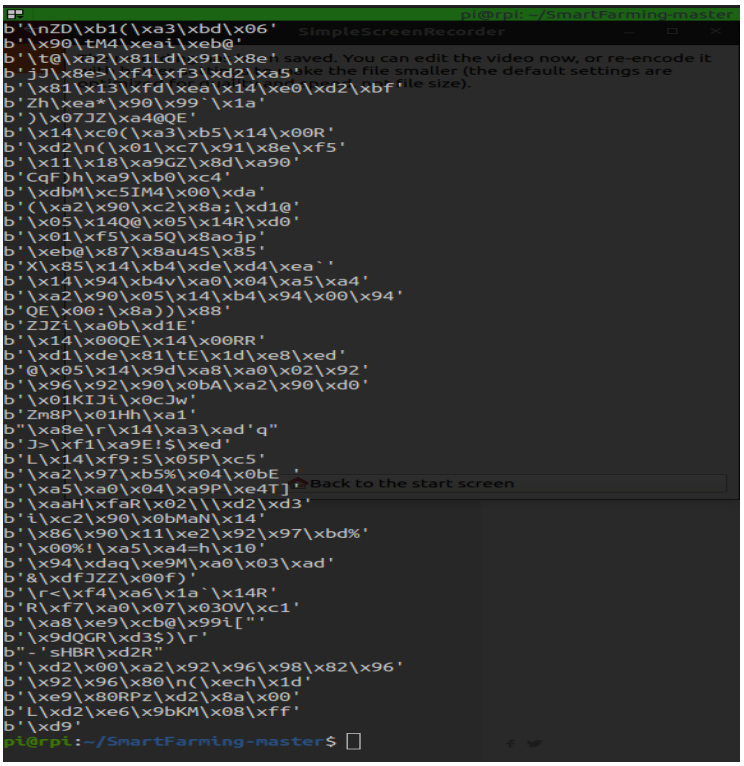

Figure 6.15: Execution and output terminal of IoT smart devices in D2G phase

6.7 Comparative study

A detailed comparative analysis among the proposed scheme (SCBAS-SF) and other existing competing authentication schemes, such as the schemes of Ali *et al.* [52], Wu and Tsai [340], Sadhukhan *et al.* [274] and Shuai *et al.* [286], is provided in this section.

6.7.1 Computation costs analysis

We use the testbed experiments with the average time on a server and a Raspberry PI 3 for cryptographic primitives as reported in Appendix B in Table B.3 for computational costs calculation for a gateway node (server) and the resource-constrained device, such as IoT smart device/sensor node, respectively. In the proposed SCBAS-SF, an IoT smart device (sensor node) SN needs $8T_h + 8T_{ecm} \approx 20.712$ ms in the D2D authentication phase and $9T_h + 4T_{ecm} \approx 11.901$ ms in the D2G authentication phase. The gateway node GWN takes $9T_h + 4T_{ecm} \approx 3.191$ ms only for in the D2G authentication phase. Table 6.5 compares the computation costs required in different schemes. The proposed SCBAS-SF has comparable computation costs in both phases as compared to other authentication schemes.

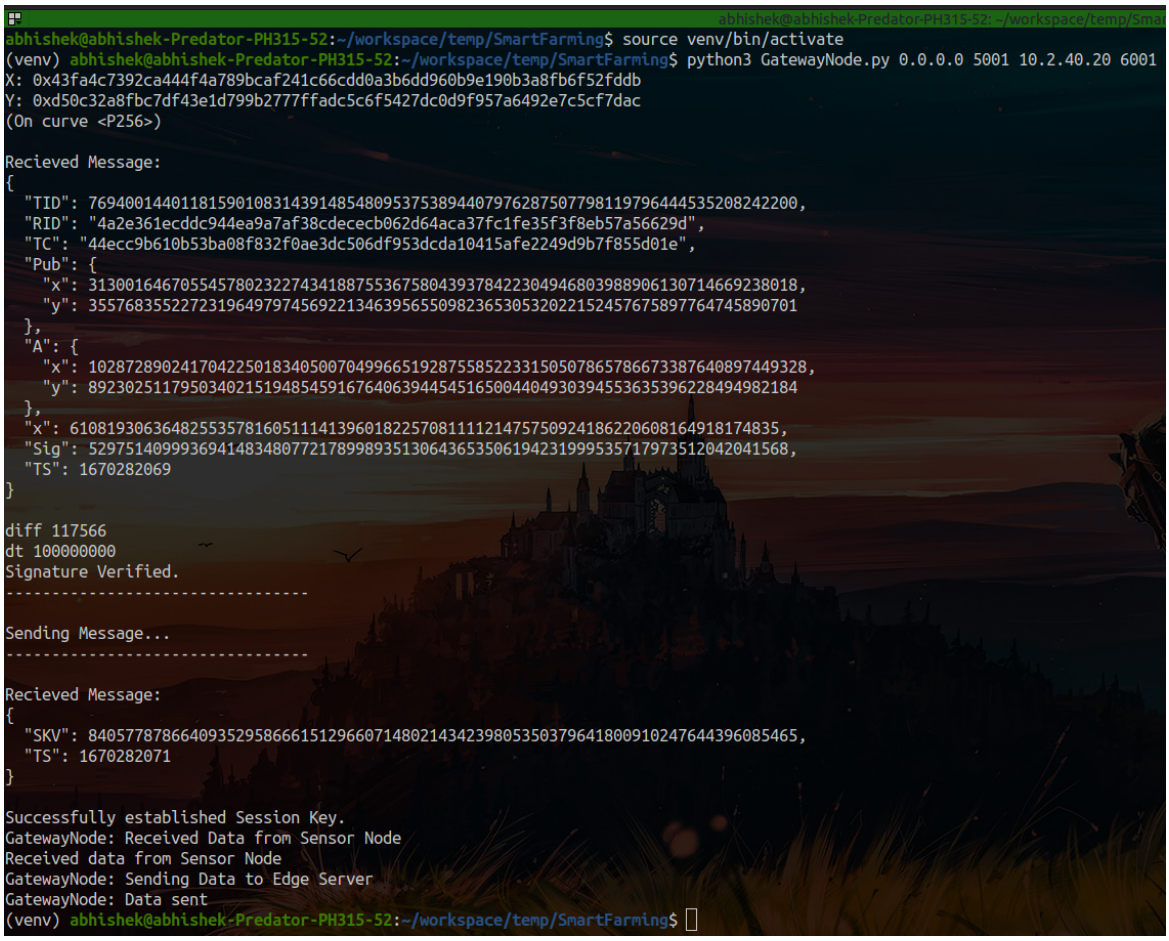

Figure 6.16: Execution and output terminal of gateway node

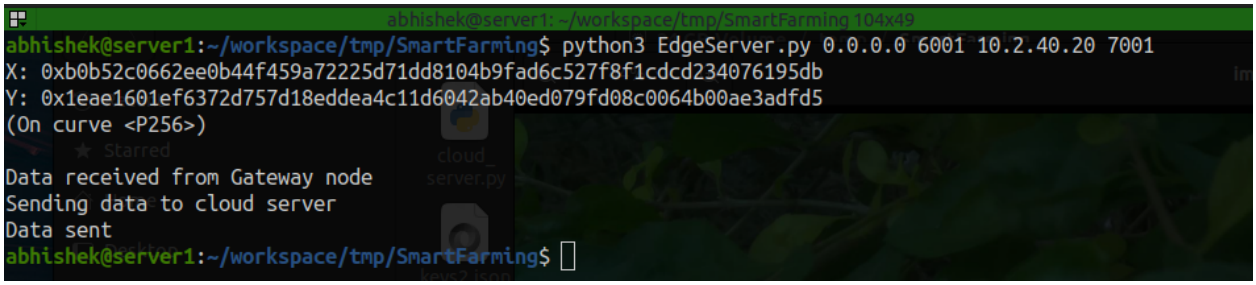

Figure 6.17: Execution and output terminal of edge server

6.7.2 Communication costs analysis

For the comparative analysis of communication costs among the proposed SCBAS-SF and other existing competing schemes, we consider the login and authentication phase. The identities and random secrets are taken to be 160 bits each. The length of the output of the

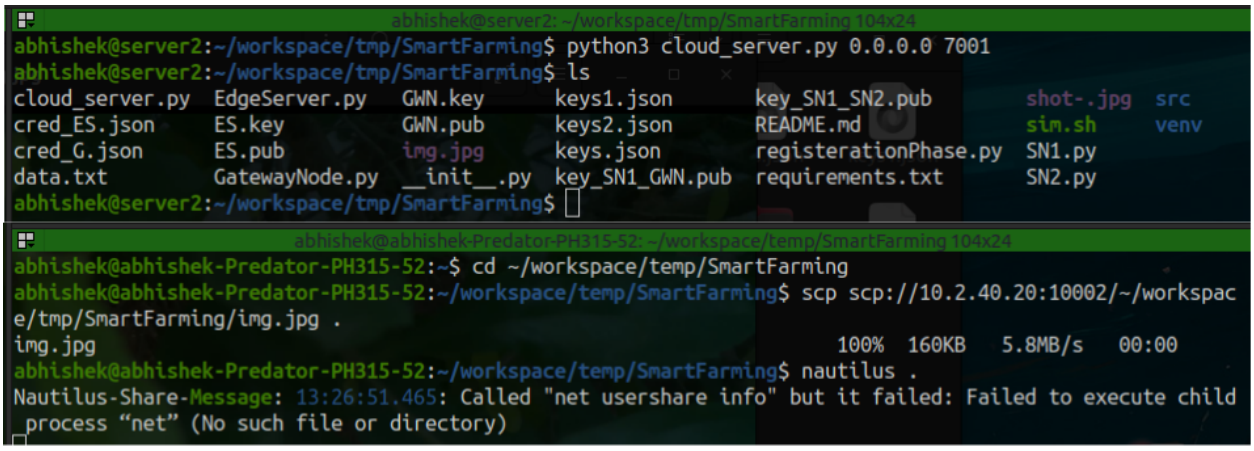

Figure 6.18: Execution and output terminal of cloud server

Figure 6.19: Image at cloud server captured from an IoT smart device through the proposed authentication scheme

hash function, the ciphertext block of "symmetric key encryption/decryption using AES-128 encryption [8]" is taken as 256 bits and 128 bits. A point $P = (x_P, y_P)$ on the elliptic curve $E_q(a, b)$ is taken as $(160 + 160) = 320$ bits, with the coordinates x_P and y_P considered as 160 bits each, assuming that 160-bit ECC provides the same security level as that for 1024-bit

Table 6.5: Comparison of computational costs

Schemes	Sensor node	Gateway node
Ali *et al.* [52]	$11T_h + T_{fe} + T_{senc} + 2T_{sdec}$ ≈ 5.733 ms	$8T_h + 5T_{enc}/T_{dec}$ ≈ 0.445 ms
Wu and Tsai [340]	$2T_{bp} + 2T_{exp} + 2T_{enc/dec} + 1T_h$ ≈ 64.965 ms	$2T_{bp} + 2T_{exp} + 2T_{enc/dec} + 1T_h$ ≈ 9.407 ms
Sadhukhan *et al.* [274]	$4T_h + 2T_{enc} + 2T_{dec} + 2T_{ecm}$ ≈ 5.876 ms	$2T_h + 2T_{dec} + 2T_{enc}$ ≈ 0.114 ms
Shuai *et al.* [286]	$13T_h + 3T_{exp}$ ≈ 4.701 ms	$7T_h + T_{exp}$ ≈ 0.457 ms
SCBAS-SF (D2D phase)	$8T_h + 8T_{ecm}$ ≈ 20.712 ms	–
SCBAS-SF (D2G phase)	$9T_h + 4T_{ecm}$ ≈ 11.933 ms	$9T_h + 4T_{ecm}$ ≈ 3.191 ms

Table 6.6: Comparison of communication overheads

Schemes	Total messages	Total cost (in bits)
Ali *et al.* [52]	5	5504
Wu and Tsai [340]	10	$1344 + 256n$
Sadhukhan *et al.* [274]	4	5248
Shuai *et al.* [286]	4	7616
SCBAS-SF (D2D phase)	3	2272
SCBAS-SF (D2G phase)	3	2304

Note: n: no. of agricultural equipment (sensor devices) in Wu and Tsai's scheme [340]

RSA public-key cryptosystem [66]. Moreover, the timestamp is taken as 32 bits.

In the proposed SCBAS-SF, during the D2D authentication phase, the messages Msg_{D2D1}, Msg_{D2D2} and Msg_{D2D3} require 992, 992 and 288 bits, respectively, which overall

Table 6.7: Comparison of security and functionality features

Features	Ali *et al.* [52]	Wu and Tsai [340]	Sadhukhan *et al.* [274]	Shuai *et al.* [286]	SCBAS-SF
$\mathcal{F}_1$	×	✓	×	✓	✓
$\mathcal{F}_2$	×	✓	×	✓	✓
$\mathcal{F}_3$	✓	NA	×	×	✓
$\mathcal{F}_4$	✓	×	×	✓	✓
$\mathcal{F}_5$	×	NA	×	N/A	✓
$\mathcal{F}_6$	✓	NA	✓	✓	✓
$\mathcal{F}_7$	×	NA	×	×	✓
$\mathcal{F}_8$	×	NA	×	×	✓
$\mathcal{F}_9$	×	×	×	×	✓
$\mathcal{F}_{10}$	×	✓	×	×	✓
$\mathcal{F}_{11}$	✓	✓	✓	✓	✓
$\mathcal{F}_{12}$	✓	✓	✓	✓	✓
$\mathcal{F}_{13}$	✓	✓	✓	✓	✓
$\mathcal{F}_{14}$	✓	✓	✓	✓	✓
$\mathcal{F}_{15}$	×	✓	×	✓	✓
$\mathcal{F}_{16}$	✓	×	✓	×	✓

Note: $\mathcal{F}_1$: "anonymity"; $\mathcal{F}_2$: "untraceability"; $\mathcal{F}_3$: "user device revocation"; $\mathcal{F}_4$: "dynamic node addition"; $\mathcal{F}_5$: "user biometric change"; $\mathcal{F}_6$: "user password change"; $\mathcal{F}_7$: "user impersonation attack"; $\mathcal{F}_8$: "stolen smart card/mobile device attack"; $\mathcal{F}_9$: "ephemeral secret leakage (ESL) attack"; $\mathcal{F}_{10}$: "privileged insider attack"; $\mathcal{F}_{11}$: "replay attack"; $\mathcal{F}_{12}$: "man-in-the-middle attack"; $\mathcal{F}_{13}$: "mutual authentication"; $\mathcal{F}_{14}$: "unauthorized login detection"; $\mathcal{F}_{15}$: "Denial-of-Service (DoS) attack"; $\mathcal{F}_{16}$: "offline guessing attacks"
N/A: "not applicable in a scheme"; ✓: "a feature is supported in a scheme or resistant against the specified attack"; ×: "a feature is not supported in a scheme or it is not resilient against the specified attack"

require 2272 bits. During the D2G authentication phase, Msg_{D2G_1}, Msg_{D2G_2} and Msg_{D2G_3} take 928, 1088, 288 bits, respectively, with a total of 2304 bits. Table 6.6 compares the communication costs in terms of number of exchanged messages and number of bits required.

The existing schemes of Ali *et al.* [52], Wu and Tsai [340], Sadhukhan *et al.* [274] and Shuai *et al.* [286] require the communication costs of 5504 bits, $1344+256n$ bits, 5248 bits and 7616 bits, respectively. It is noticed that the proposed SCBAS-SF requires low communication cost as compared to those for other schemes.

6.7.3 Security and functionality features analysis

Table 6.7 shows the comparative analysis on various "security and functionality features" ($\mathcal{F}_1$–$\mathcal{F}_{16}$) of the proposed SCBAS-SF with the schemes of Ali *et al.* [52], Wu and Tsai [340], Sadhukhan *et al.* [274] and Shuai *et al.* [286]. It shows that SCBAS-SF provides superior security and more functionality features as compared to all compared authentication schemes.

6.8 Summary

This chapter proposes a new authentication using smart contracts for a smart farming architecture that uses a hybrid blockchain in conjunction with edge computing. The proposed scheme (SCBAS-SF) supports security features of anonymity and traceability. SCBAS-SF is shown to be provably secure using the widely recognized ROR model. Furthermore, it is shown to be resilient against a number of potential attacks using informal security analysis. In addition, SCBAS-SF is also formally verified against replay and man-in-the-middle attacks using the AVISPA tool. The practical implementation of the proposed SCBAS-SF is shown on the hyper ledger sawtooth platform and also on the MIRACL-based Raspberry PI 3 testbed. A real-time testbed has been implemented to run the the proposed protocol and the configurations along with the output has been explained in detail. A thorough comparative analysis shows the proposed SCBAS-SF has superior security with comparable computation cost and low communication cost as compared to other existing competing authentication schemes.

Chapter 7

Blockchain-Enabled Authenticated Key Agreement for Mobile Vehicles Assisted Precision Agricultural IoT Networks

Precision farming has a positive potential in the agricultural industry regarding water conservation, increased productivity, better development of rural areas, and increased income. Blockchain technology is a better alternative for storing and sharing farm data as it is reliable, transparent, immutable, and decentralized. Remote monitoring of an agricultural field requires security systems to ensure that any sensitive information is exchanged only among authenticated entities in the network.

An active blockchain stores and retrieves both data and secret credentials simultaneously without the deployment of smart contracts, while the registration and authentication are in progress. The transaction contents in a block from an active blockchain have secret credentials that are required to execute the cryptographic operations in the authentication. Without these credentials, the authentication process would be unable to execute all its steps, and it will lead to a failed authentication. An inactive or passive blockchain is accessed independently of authentication. It does not affect the outcome of authentication between the entities. The full potential of a passive blockchain is not harnessed as its use is limited to one phase and stores either credentials or data but not both.

Most existing authentication schemes in smart agriculture [52, 54, 58, 88, 96, 102, 265] do not use blockchain technology. The scheme in [340] uses the blockchain technology, but

it is inefficient due to its high cost. The previous work carried out in Chapter 6 proposed an authentication scheme for smart agriculture using hybrid blockchain with smart contracts. Here, blockchain has been used to store sensor data securely after the authentication process between the involved entities is completed. Our other previous work in Chapter 5 proposed the use of private blockchain in agriculture environment in conjunction with unmanned aerial vehicles (UAV). However, it can store only classified data specific to a part of the stakeholder community. Neither of these works use the concept of an active blockchain or the use of mobile agricultural vehicles. However, all these use the blockchain passively only to store the data from the sensing equipment in the IoT networks. Thus, the blockchain does not play any role during the authentication process. Moreover, no existing works have attempted to use the active blockchain alongside elliptic curve operations. None of the schemes have used agricultural vehicles in their model. Thus, no current schemes have considered using elliptic curve cryptography (ECC) together with mobile vehicles and hybrid blockchain to achieve mutual authentication in smart agriculture.

A smart agriculture environment uses several vehicles such as tractors, harvesters, farm trucks, bale handlers, balers, crop sprayers, front-end loaders, lawn mowers, rollers, cultivators, harrows, subsoilers, seed drills, land imprinter, stone picker, manure spreader/honey wagon, tree shaker, swather, and several other machines. These machines may be manually driven or operated autonomously. The idea for this work stems from the fact that vehicles intrinsic to the farming process should be put to good use for secure data transmission in a smart farm. The role of blockchain in the current work is to store secret credentials for authentication along with sensor data. Only part of the data collected in smart agriculture needs to be encrypted while the rest of the data resides unencrypted. Therefore, hybrid blockchain is most appropriate for smart agriculture. Each block in the proposed hybrid blockchain consists of partly encrypted transactions to store credentials or fully encrypted transactions to store sensor data. It is well-known that ECC supports design of lightweight schemes. Hence, ECC involving hybrid blockchain over fog servers with mobile vehicles as data collectors has a valuable potential to achieve mutual authentication in smart agriculture.

The novel contribution of the blockchain part in the proposed scheme lies in the increased potential use of blockchain in multiple phases to store data and credentials together. Access to the blockchain is inevitable during authentication as it holds critical credentials from registration. It has the advantage of avoiding the privileged-insider attack, which is an essential attack in any authenticated key agreement scheme. In addition, the blockchain is also used for storing sensor data rather than storing it in semi-trusted cloud servers. If we

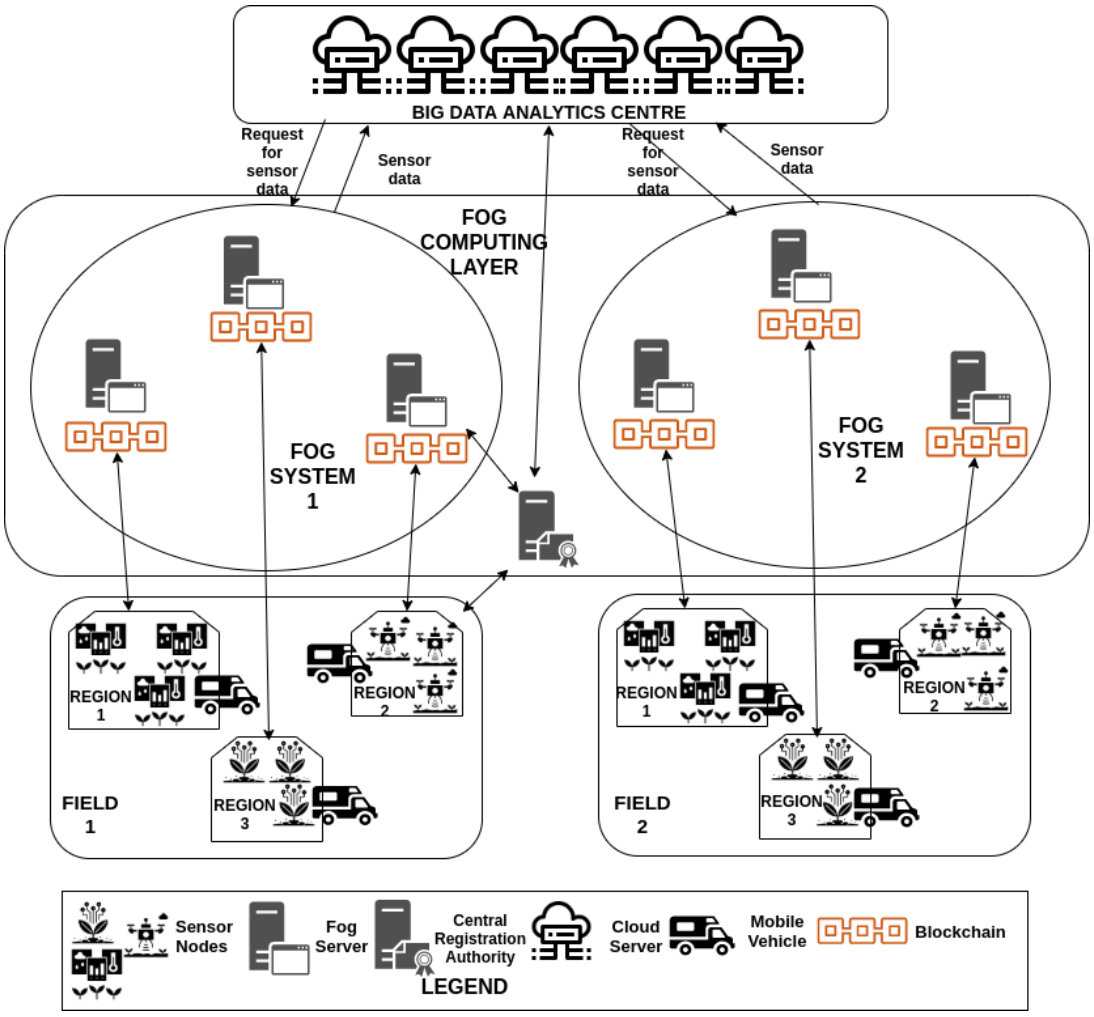

Figure 7.1: Blockchain-based mobile vehicles-assisted precision agricultural IoT network

keep the data in semi-trusted cloud servers, there are possibilities of data poisoning attacks that are very crucial concerns, and they may cause a significant factor for the businesses and organizations for both financial terms as well as damaging their reputations when the Big data analytics are performed on the analyzed data which becomes corrupted [242].

7.1 Network model

The proposed scheme is designed for a network model which is presented in Figure 7.1. This architecture applies to many agricultural fields where each field is divided into disjoint regions. Hundreds of sensor nodes SN are scattered across the field to collect environmental

readings. Various mobile vehicles MV are used for farming activities in agricultural field work as mobile sinks to collect data from the sensors in that field. The collected data is then sent to the fog computing layer. Each agricultural region is assigned to a fog server FS. The fog servers connected to the regions in a single agricultural field form a fog system to maintain a decentralized blockchain that can store and process data required during the authentication process, along with running a consensus algorithm. The proposed scheme uses the blockchain actively during the authentication process for managing critical parameters, along with passive data storage from the sensors after the authentication is completed.

The considered blockchain is a consortium blockchain that can have two types of blocks: a) AuthCred block, which stores the credentials needed during the authentication process, and b) SensorData block, which stores the sensor data received after encryption with the key established during the key management process. There are two types of AuthCred blocks, one is for mobile vehicles and the other is for the fog servers. For the sensor data to be securely stored in the blockchain, the first authentication is required between the sensors and the mobile vehicle. The second authentication is required between the mobile vehicle and the fog server, which takes credentials from the blockchain to verify the requesting vehicle. The sensor data is then forwarded from the sensor to the fog server via the mobile vehicle and stored on the blockchain. When the Big Data analytics center (BDAC) requires sensor data, it needs to send a request to the appropriate fog server, which retrieves the requested sensor data from the blockchain, encrypts it with the public key of the cloud storage inside BDAC, and sends it as a response to BDAC.

7.2 Threat model

For a systematic analysis of the required defenses for the proposed authentication scheme, the standard "Dolev-Yao (DY) threat model" [125] and the current *de facto* "Canetti and Krawczyk's model (CK-adversary model)" [94] have been considered as the most appropriate threat models. An adversary $\mathcal{A}$ under the DY threat model can perform the following actions on the network model:

- All communication in public channels among the smart devices, mobile vehicles, fog servers, and cloud servers is accessible to $\mathcal{A}$ and allows it to seize, remove, modify and re-transmit existing messages or circulate counterfeit messages.
- $\mathcal{A}$ may impersonate a smart device, mobile vehicle, or fog server, and carry out tasks on their behalf.

• $\mathcal{A}$ may initialize multiple executions of the protocol simultaneously. Smart devices, mobile vehicles, fog servers, and cloud servers may take part in any number of such concurrent executions at the same time.

• The smart devices, mobile vehicles, fog servers, and cloud servers are honest and stateless, whereas $\mathcal{A}$ is stateful.

An adversary $\mathcal{A}$ under the CK-adversary threat model can perform the following actions on the network model:

• $\mathcal{A}$ enjoys all the capabilities as in the DY threat model.

• $\mathcal{A}$ can extract secret credentials by hijacking session states during communication among the communicating smart devices, mobile vehicles, fog servers, and cloud servers.

In addition, we assume that $\mathcal{A}$ can physically capture some smart devices as well as mobile vehicles to extract secret credentials from their memory using the power analysis attacks [235] and timing attacks [199]. The fog servers and cloud servers are assumed to be under a physical locking system as suggested in [75, 335]. Thus, it is assumed that $\mathcal{A}$ cannot launch stolen verifier attacks on fog servers and cloud servers as all secret credentials stored on these servers are placed in their secure databases.

7.3 Research contributions

The novel contributions of this chapter are summarized below.

- The blockchain is leveraged to its full potential by using it in multiple phases. Specifically, we propose a new blockchain-enabled authenticated key agreement scheme for mobile vehicles-assisted precision agricultural IoT networks, called *AgroMobiBlock*, which makes use of an active consortium blockchain. To the best of our knowledge, this is the first attempt to use an active blockchain alongside elliptic curve operations in smart farming.

- The proposed scheme leverages vehicular farming systems during the authentication process, which has not been explored in the existing farming applications.

- A real-time implementation using testbed setup gives the step-wise execution time of each phase of *AgroMobiBlock*. In addition, the blockchain simulation observes that the consensus time has a significant increase with the number of nodes and a small increase with the number of transactions. Increasing the number of transactions per

block increases the throughput, but it reduces the service time of the blockchain as well.

7.4 Proposed scheme

In this section, we describe a new blockchain-enabled authenticated key agreement scheme for mobile vehicles-assisted precision agricultural IoT networks, called $AgroMobiBlock$. Various notations that are used in this phase are provided in Table 7.1 with their descriptions.

7.4.1 High-level protocol overview

The proposed scheme begins with a one-time registration of the involved entities of IoT smart devices, mobile vehicles, and fog servers. The registration of the mobile vehicle and fog server creates an AuthCred block, which stores the authentication credentials to be used during the authentication phase between a mobile vehicle and a fog server ($MVFS$). Part of each transaction in such a block is encrypted. There are two types of AuthCred blocks: 1) one is for storing private parameters for a fog server and 2) other is for storing private parameters for a mobile vehicle.

The first phase of authentication between an IoT smart device and a mobile vehicle ($SNMV$) is required between a sensor node and a mobile vehicle culminating in a session key agreement where the session key consists of a private hash from the sensor node and a private hash from the mobile vehicle. The second phase of authentication between a mobile vehicle and a fog server ($MVFS$) retrieves the long-term secret credentials from the blockchain to verify the requesting vehicle. A session key is then established, which consists of a long-term secret from a mobile vehicle encrypted using an association key, a private hash from the mobile vehicle, a private hash from the fog server, and a long-term secret from the fog server encrypted using the association key, and the Diffie-Hellman type key.

The sensor data is forwarded from the sensors to the fog server via the mobile vehicle using the established session keys in the $SNMV$ and $MVFS$ phases. The fog server creates a transaction out of the received sensor data and a SensorData block, out of a collection of such transactions. This block is then added to the single hybrid blockchain. To achieve the requirements that Basin *et al.* [68] proposed for entity authentication, each message includes the temporary and pseudo-identity of the sender, and the trusted registration authority (TRA) uniquely carries out the role of trusted authority.

Table 7.1: Notations and their descriptions

Notation	Significance
$E_q(\kappa, \mu)$	A non-singular elliptic curve of the form: $y^2 = x^3 + \kappa x + \mu \pmod q$ over Galois field $GF(q)$
G	A base point in $E_q(\kappa, \mu)$ of order is n_G as big as q
$x \cdot G$	Elliptic curve point multiplication: $x \cdot G = G + G + \cdots + G\,(x\,times)$
$A + B$	Elliptic curve point addition; $A, B \in E_q(\kappa, \mu)$
TRA	Trusted Registration Authority
SN	IoT smart device
MV	Mobile vehicle
FS	Fog server
$BDAC$	Big data analytics center
RTS_X	Registration timestamp issued by the TRA to entity X
ID_M, TID_M, RID_M	MV's real identity, temporary identity, and pseudo-identity, respectively
ID_F, TID_F, RID_F	FS's real identity, temporary identity, and pseudo-identity, respectively
ID_{SND}, TID_{SND}, RID_{SND}	SN's real identity, temporary identity, and pseudo-identity, respectively
pr_{TRA}, Pub_{TRA}	Private and public key of TRA, respectively
pr_M, Pub_M	Private key and public key of MV, respectively
pr_F, Pub_F	Private key and public key of FS, respectively
pr_S, Pub_S	Private key and public key of SN, respectively
K_{MV_i,FS_j}	Association key between MV_i and FS_j
m, j_m, u_m	MV's random secrets
f, v_f	FS's random secrets
s, i_s	SN's random secrets
$\|\|$	Concatenation operation
TS_X	Current timestamp produced by an entity X
$*$	Integer multiplication operation
$\oplus$	Exclusive OR operation
ΔT	Maximum allowed delay in transmission for a particular message
$H(\cdot)$	"Collision-resistant cryptographic one-way hash function"

The blockchain is truly hybrid in the sense that it consists of two types of blocks: 1) AuthCred block, which contain the registration credentials needed during the authentication process, and 2) SensorData block, which contains the sensor data received securely during the authenticated key management process. Each transaction in the AuthCred block consists of partly encrypted and partly unencrypted contents. Critical content is first encrypted, com-

bined with unencrypted content, and a transaction is created from both the unencrypted and encrypted contents. The transactions in the SensorData block consist of only unencrypted content inside. Encryption is applied to the entire transaction, and a collection of such encrypted transactions are made into blocks. Two different types of blocks are stored in a single hybrid blockchain, which reduces the overhead of using separate blockchains for storing authentication credentials and sensor data. Two different types of blocks for the separate storage of authentication credentials and sensor data help in keeping a clear distinction of access control between the two. Any access to sensor data will not reveal the authentication credentials used in any session and vice-versa. Hence, different levels of access control may be applied to them in the future.

Private channels are established directly between the TRA and the entities (SN, MV, and FS). There are no private channels among the entities SN, MV, and FS. These private channels are used during the registration phase as described in Section 7.5.2 to send and receive private identities, private random secrets, and private session key parameters that should be accessible only to TRA and the registering entity. Public channels are established separately between the entities SN and MV, and the entities MV and FS. The messages communicated during the authentication process in Section 7.4.4 and secure data aggregation phase in Section 7.4.5 use these public channels. A symbolic representation of the proposed $AgriMobiBlock$ is shown in Figure 7.2.

The proposed authentication and key agreement protocol ($AgroMobiBlock$) consists of the following phases as described below in detail.

7.4.2 System initialization phase

This phase involves the following steps:

- **Step** S_1: The Trusted Registration Authority (TRA) selects a non-singular elliptic curve $E_q(\kappa, \mu) : y^2 = x^3 + \kappa x + \mu \pmod q$ over the Galois field $GF(q)$, with a "point at infinity (zero point)" $\mathcal{O}$, constants $\kappa, \mu \in Z_q = \{0, 1, 2, \cdots, q-1\}$ such that $4\kappa^3 + 27\mu^2 \neq 0 \pmod q$ is satisfied. The TRA picks a base point $G \in E_q(\kappa, \mu)$ whose order n_G as large as q, that is, $n_G \cdot G = G + G + \cdots + G$ (n_G times) $= \mathcal{O}$, the point at infinity or zero point.

- **Step** S_2: The TRA picks a "collision-resistant one-way cryptographic hash function", say $H(\cdot)$ (for instance, "Secure Hash Standard (SHA-256) hash algorithm may be used).

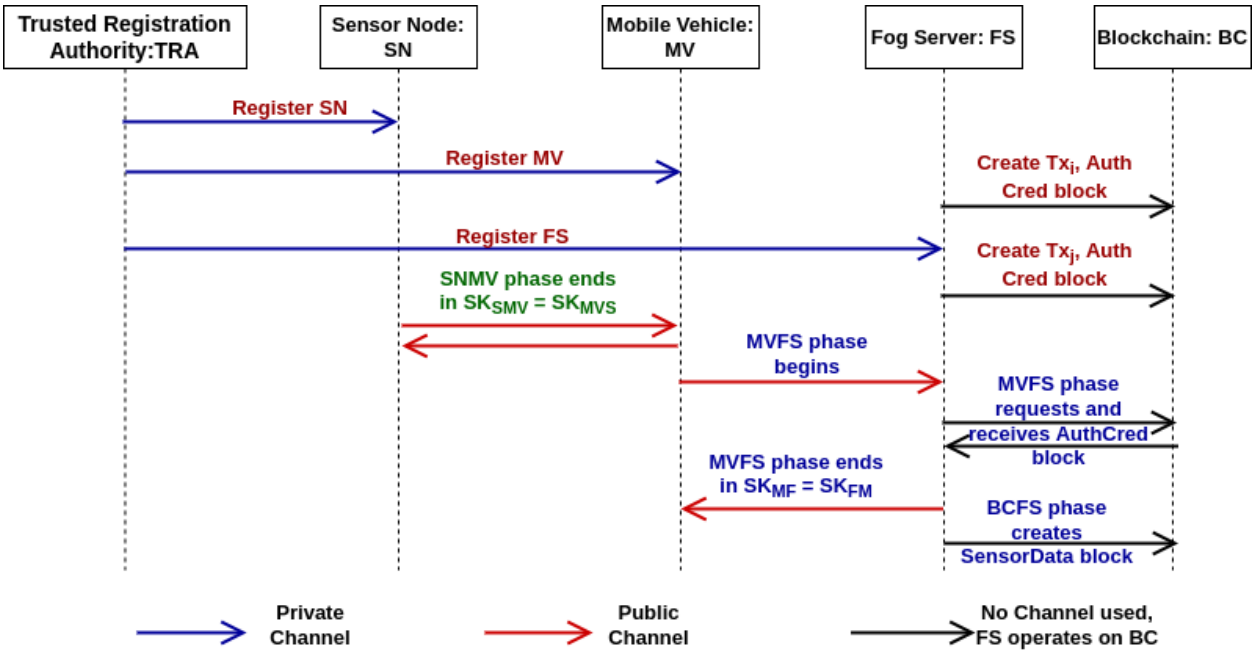

Figure 7.2: Overview of the proposed *AgroMobiBlock*

- **Step** S_3: The TRA chooses a private key pr_{TRA} in $Z_q^* = \{1, 2, \cdots, q-1\}$ for itself and computes the public key $Pub_{TRA} = pr_{TRA}.\ G$, and publishes Pub_{TRA} and domain parameters $\{E_q(\kappa, \mu),\ G,\ H(\cdot)\}$ as public.

7.4.3 Registration phase

The TRA executes this phase to register each entity individually through a dedicated registration phase.

1) **IoT smart device registration phase**: This phase allows the TRA to register the IoT Smart Sensor (SN) using secure channel.

   - **Step** SDR_1**:** TRA picks ID_{SND}, $s \in Z_q^*$, RTS_S and computes the pseudo-identity as $RID_{SND} = H(ID_{SND}||\ s||\ RTS_S||\ pr_{TRA})$ and the temporary identity as $TID_{SND} = H(RID_{SND}||\ s||\ pr_{TRA}||\ RTS_S)$. TRA picks the private key as $pr_S \in Z_q^*$ and the corresponding public key as $Pub_S = pr_S \cdot G$.
   - **Step** SDR_2**:** TRA pre-loads SN with $\{(RID_{SND},\ TID_{SND}),\ H(\cdot),\ E_q(\kappa, \mu),\ G,\ (pr_S,\ Pub_S)\}$.

IoT Smart Device Registration Phase	
Trusted Registration Authority (TRA)	IoT Smart Device (SN)
TRA picks ID_{SND}, $s \in Z_q^*$, RTS_S	
$RID_{SND} = H(ID_{SND}\|\| s\|\| RTS_S\|\| pr_{TRA})$	
$TID_{SND} = H(RID_{SND}\|\| s\|\| pr_{TRA}\|\| RTS_S)$	
TRA picks $pr_S \in Z_q^*$	
$Pub_S = pr_S \cdot G$.	
TRA preloads SN with $\{(RID_{SND}, TID_{SND}), H(\cdot), E_q(\kappa, \mu), G, (pr_S, Pub_S)\}$	SN stores $\{(RID_{SND}, TID_{SND}, K_S), H(\cdot), E_q(\kappa, \mu), G, (pr_S, Pub_S)\}$ in its memory

Figure 7.3: Summary of IoT Smart Device Registration Phase

2) **Mobile vehicle registration phase**: This phase allows the TRA to register the Mobile Vehicle (MV) using a secure channel.

- **Step** MVR_1**:** MV picks its private identity ID_M and forwards it to the TRA via a secure channel. TRA picks a private random secret $m \in Z_q^*$ and generates a timestamp for identity generation in registration RTS_m. TRA then computes the pseudo-identity $RID_M = H(ID_M\|\| m\|\| RTS_m\|\| pr_{TRA})$ and the temporary identity $TID_M = H(RID_M\|\| m\|\| pr_{TRA}\|\| RTS_m)$. TRA sends RID_M, TID_M back to the MV via the secure channel.
- **Step** MVR_2**:** MV picks its private key as pr_M and the corresponding public key as $Pub_M = pr_M \cdot G$. The mobile vehicle then publishes Pub_M as its public key and sends Pub_M to TRA via the secure channel.
- **Step** MVR_3**:** TRA generates a private session key parameter for MV as $K_M = H(RID_M\|\| Pub_M\|\| pr_{TRA}\|\| ID_M\|\|m)$. TRA then associates the MV with its FS by generating an association key K_{MV_i,FS_j} or retrieving the association key from the blockchain if it already exists. It also generates a timestamp TS_{mc} and computes $K_M^* = H(K_M\|\| TS_{mc}) \oplus H(K_{MV_i,FS_j}\|\| TID_M\|\| RID_M\|\|TS_{mc})$ that hides the MV's contribution to the session key with FS.
- **Step** MVR_4**:** TRA creates a transaction $Tx_i = \langle TID_M, K_M^*, E_{Pub_M}(ID_M, K_{MV_i,FS_j}, RID_M, TS_{mc})\rangle$ and signs the transaction with $Sig_{Tx_i} = ECDSA.sig_{pr_{TRA}}(Tx_i)$ using the signature method $sig(\cdot)$ of the "elliptic curve digital signature algorithm (ECDSA)" with the private key pr_{TRA}. After that TRA

forwards $\langle Tx_i, Sig_{Tx_i} \rangle$ to the fog server FS via a secure channel, which stores Tx_i in its local memory after the signature Sig_{Tx_i} is verified. The leader fog server creates AuthCred block with its collected n_m transactions, executes the "Practical Byzantine Fault Tolerance (PBFT)" described in [95] (see Algorithm 4) within the same fog system, followed by mining and addition of the block into the blockchain. An MV is associated with a single FS only.

Mobile Vehicle Registration Phase		
Mobile Vehicle (MV)	Central Registration Authority (TRA)	Fog Server (FS)
MV picks ID_M		
$\xrightarrow{ID_M}$ (Secure Channel)		
	TRA picks m, RTS_m	
	$RID_M = H(ID_M \|\| m \|\| RTS_m \|\| pr_{TRA})$	
	$TID_M = H(RID_M \|\| m \|\| pr_{TRA} \|\| RTS_m)$	
	$\xleftarrow{RID_M, TID_M}$ (Secure Channel)	
MV picks pr_M		
$Pub_M = pr_M \cdot G$		
$\xrightarrow{Pub_M}$ (Secure Channel)		
	$K_M = H(RID_M \|\| Pub_M \|\| pr_{TRA} \|\| ID_M \| m)$	
	TRA retrieves or generates K_{MV_i,FS_j}, generates TS_{mc}	
	$K_M^* = H(K_M \| TS_{mc}) \oplus$ $H(K_{MV_i,FS_j} \|\| TID_M \|\| RID_M \| TS_{mc})$	
	TRA creates $Tx_i = \langle TID_M, K_M^*,$ $E_{Pub_M}(ID_M, K_{MV_i,FS_j}, RID_M, TS_{mc}) \rangle$	
	TRA creates $Sig_{Tx_i} = Sig_{pr_{TRA}}(H(Tx_i))$	
	$\xrightarrow{\langle Tx_i, Sig_{Tx_i} \rangle}$ (Secure Channel)	
		Verify $Sig_{Tx_i} \stackrel{?}{=} Sig_{Pub_{TRA}}(H(Tx_i))$
		If so, Store TX_i in memory
		Create AuthCred block

Figure 7.4: Summary of mobile vehicle registration phase

3) **Fog server registration phase**: This phase allows the TRA to register a fog server (FS).

- **Step FSR_1:** FS picks its private identity as ID_F and forwards it to the TRA via a secure channel. TRA picks its private random secret $f \in Z_q^*$ and generates a timestamp RTS_f. TRA then computes the pseudo-identity $RID_F = H(ID_F \|\| f \|\|$

$RTS_f || pr_{TRA}$) and the temporary identity $TID_F = H(RID_F || f || pr_{TRA} || RTS_f)$. TRA sends RID_F, TID_F back to the mobile vehicle MV via the secure channel.

- **Step FSR_2:** FS picks its private key pr_F and the corresponding public key as $Pub_F = pr_F \cdot G$. The FS then publishes Pub_F as its public key and sends Pub_F to TRA via a secure channel.
- **Step FSR_3:** TRA generates a private session key parameter for FS as $K_F = H(RID_F || Pub_F || pr_{TRA} || ID_F || f)$. TRA then associates the FS with all its MV by generating an association key K_{MV_i,FS_j} or retrieving the association key from the blockchain if it already exists. It also generates a timestamp TS_{fc} and computes $K_F^* = H(K_F || TS_{fc}) \oplus H(K_{MV_i,FS_j} || TID_F || RID_F || TS_{fc})$ that hides the FS's contribution to the session key with MV.
- **Step FSR_4:** TRA creates a transaction $Tx_j = \langle TID_F, K_F^*, E_{Pub_F}(ID_F, K_{MV_i,FS_j}, RID_F, TS_{fc}) \rangle$ and signs this transaction with its private key pr_{TRA} to obtain $Sig_{Tx_j} = ECDSA.sig_{pr_{TRA}}(Tx_j)$. Next, TRA forwards $\langle Tx_j, Sig_{Tx_j} \rangle$ to a fog server in the blockchain for mining. The leader fog server creates an AuthCred block with its collected n_f transactions, executes the consensus algorithm (see Algorithm 4) within the same fog system, followed by mining and adds the block to the blockchain. An FS may be associated with multiple MVs, and the different association keys for a given FS are identified by the corresponding TID_M of the MV as stored in the transaction.

7.4.4 Authentication phase

In this section, we consider two types of authentication mechanisms: 1) between an IoT smart device (SN) and a mobile vehicle (MV), called SNMV authentication and 2) between a mobile vehicle (MV) and a fog server (FS), called MVFS authentication.

1) **Authentication between IoT smart device and mobile vehicle**: The following steps explain this phase (SNMV authentication phase):

- **Step $SNMV_1$:** SN picks a private random secret $i_S \in Z_q^*$ along with a timestamp TS_S. It computes the corresponding public parameter as $I_S = H(i_S || TID_{SND} || RID_{SND} || pr_S || TS_S) \cdot G$ and a signature on the message as $Sig_S = H(i_S || TID_{SND} || RID_{SND} || pr_S || TS_S) + H(RID_{SND} || Pub_S || TID_{SND} || TS_S) * pr_S \pmod q$. The

Fog Server Registration Phase	
Fog Server (FS)	Central Registration Authority (TRA)
FS picks ID_F	
$\xrightarrow{ID_F}$ (Secure Channel)	
	TRA picks f, RTS_f
	$RID_F = H(ID_F \|\| f \|\| RTS_f \|\| pr_{TRA})$
	$TID_F = H(RID_M \|\| f \|\| pr_{TRA} \|\| RTS_f)$
	$\xleftarrow{RID_F, TID_F}$ (Secure Channel)
FS picks pr_F	
$Pub_F = pr_F \cdot G$	
$\xrightarrow{Pub_F}$ (Secure Channel)	
	$K_F = H(RID_F \|\| Pub_F \|\| pr_{TRA} \|\| ID_F \| \| f)$
	TRA retrieves or generates K_{MV_i,FS_j}, generates TS_{fc}
	$K_F^* = H(K_F \| \| TS_{fc}) \oplus$ $H(K_{MV_i,FS_j} \|\| TID_F \|\| RID_F \| \| TS_{fc})$
	TRA creates $Tx_j = \langle TID_F, K_F^*,$ $E_{Pub_F}(ID_F, K_{MV_i,FS_j}, RID_F, TS_{fc}) \rangle$
	TRA creates $Sig_{Tx_j} = Sig_{pr_{TRA}}(H(Tx_j))$
	$\xleftarrow{\langle Tx_j, Sig_{Tx_j} \rangle}$ (Secure Channel)
Verify $Sig_{Tx_j} \stackrel{?}{=} Sig_{Pub_{TRA}}(H(Tx_j))$	
If so, Store TX_j in memory	
Create AuthCred block	

Figure 7.5: Summary of fog server registration phase

sensor SN sends the message Msg_{SM_1}: $\langle I_S, TID_{SND}, RID_{SND}, TS_S, Sig_S \rangle$ to the mobile vehicle MV.

- **Step $SNMV_2$:** The mobile vehicle MV receives the message Msg_{SM_1} at the time TS_S^* and verifies the timestamp as $|TS_S^* - TS_S| \leq \Delta$T. If it is verified as true, the signature Sig_S is verified. The signature is verified as $Sig_S \cdot G \stackrel{?}{=} I_S + H(RID_{SND} \|\|$

$Pub_S||\ TID_{SND}||\ TS_S) \cdot Pub_S$.

- **Step** $SNMV_3$**:** MV picks a private random secret $j_M \in Z_q^*$ and a timestamp TS_M to compute the public parameter $J_M = H(j_M||\ TID_M||\ RID_M||\ pr_M||\ TS_M) \cdot G$. The session key between the sensor and the mobile vehicle is computed as $SK_{MVS} = H(j_M||\ TID_M||\ RID_M||\ pr_M||\ TS_M) \cdot I_S$. A signature is generated over the private random j_M and the session key as $Sig_M = H(j_M||\ TID_M||\ RID_M||\ pr_M||\ TS_M) + H(J_M||\ SK_{MVS}||\ TID_{SND}||\ TS_M)\ *\ pr_M \pmod q$. MV picks a new temporary session identity for the smart device $TID_{SND}^{new} \in Z_q^*$ and hides it by computing $TID_{SND}^* = TID_{SND}^{new} \oplus H(TID_{SND}||\ SK_{MVS}||\ TS_M||\ Sig_M)$. MV sends Msg_{SM_2} : $\langle J_M, Sig_M, TID_M, RID_M, TID_{SND}^*, TS_M\rangle$ to SN.
- **Step** $SNMV_4$**:** SN receives the message Msg_{SM_2} at the timestamp TS_M^* and verifies it as $|TS_M^* - TS_M| \leq \Delta$T. It then computes the session key as $SK_{SMV} = H(i_S||\ TID_{SND}||\ RID_{SND}||\ pr_S||\ TS_S) \cdot J_M$ and checks the signature Sig_M as $Sig_M \cdot G \stackrel{?}{=} J_M + H(J_M||\ SK_{SMV}||\ TID_{SND}||\ TS_M) \cdot Pub_M$. If the signature verification is successful, then it extracts $TID_{SND}^{new} = TID_{SND}^* \oplus H(TID_{SND}||\ SK_{SMV}||\ TS_M||\ Sig_M)$ and updates its current temporary identity TID_{SND} with new temporary identity TID_{SND}^{new}. It then generates a new temporary identity for MV as $TID_M^{new} \in Z_q^*$ and a timestamp TS_{SM}. It hides TID_M^{new} as $TID_M^* = TID_M^{new} \oplus H(TID_M||\ SK_{SMV}||\ TS_{SM})$ and computes the session key verifier as $SKV_{SMV} = H(SK_{SMV}||\ TS_{SM}||\ TID_M^{new})$ and sends the message Msg_{SM_3}: $\langle TID_M^*, SKV_{SMV}, TS_{SM}\rangle$ to the mobile vehicle.
- **Step** $SNMV_5$**:** After the timestamp is verified correctly as $|TS_M^* - TS_M| \leq \Delta$T using Msg_{SM_2} received at TS_M^*, MV extracts TID_M^{new} from $TID_M^{new} = TID_M^* \oplus H(TID_M||\ SK_{MVS}||\ TS_{SM})$ and computes the session key verifier as $SKV_{MVS} = H(SK_{MVS}||\ TS_{SM}||\ TID_M^{new})$. If the received SKV_{SMV} is equal to the computed SKV_{MVS}, it stores the session key SK_{MVS} and updates TID_M with TID_M^{new}. SN also stores the session key SK_{SMV} in its memory for the later communication with MV.

This phase is summarized in Figure 7.6.

2) **Authentication between mobile vehicle and fog server**: This MVFS authentication phase establishes a session key between a mobile vehicle (MV) and its associated fog server (FS) after mutual authentication upon retrieving the registered credentials from the blockchain during registration process.

IoT Smart Sensor Device (SN)	Mobile Vehicle (MV)
Pick $i_S \in Z_q^*$, timestamp TS_S.	
Compute	
$I_S = H(i_S \|\| TID_{SND} \|\| RID_{SND} \|\| pr_S \|\| TS_S) \cdot G$,	Check if $\|TS_S^* - TS_S\| \leq \Delta$T? If so, verify
$Sig_S = H(i_S \|\| TID_{SND} \|\| RID_{SND} \|\| pr_S \|\| TS_S) +$	$Sig_S \cdot G \stackrel{?}{=} I_S + H(RID_{SND} \|\| Pub_S \|\|$
$H(RID_{SND} \|\| Pub_S \|\| TID_{SND} \|\| TS_S)$ * $pr_S \pmod q$.	$TID_{SND} \|\| TS_S) \cdot Pub_S$. If valid,
$\underrightarrow{Msg_{SM_1} : \langle I_S, TID_{SND}, RID_{SND}, TS_S, Sig_S \rangle}$	pick $j_M \in Z_q^*$, timestamp TS_M. Compute
	$J_M = H(j_M \|\| TID_M \|\| RID_M \|\| pr_M \|\| TS_M) \cdot G$,
	$SK_{MVS} = H(j_M \|\| TID_M \|\| RID_M \|\| pr_M \|\| TS_M) \cdot I_S$,
	$Sig_M = H(j_M \|\| TID_M \|\| RID_M \|\| pr_M \|\| TS_M)$
	$+ H(J_M \|\| SK_{MVS} \|\| TID_{SND} \|\| TS_M)$ * $pr_M \pmod q$.
Check if $\|TS_M^* - TS_M\| \leq \Delta$T? If so, compute	Pick $TID_{SND}^{new} \in Z_q^*$. Compute $SK_{SMV} =$
$H(i_S \|\| TID_{SND} \|\| RID_{SND} \|\| pr_S \|\| TS_S) \cdot J_M$.	$TID_{SND}^* = TID_{SND}^{new} \oplus H(TID_{SND} \|\|$
Verify $Sig_M \cdot G \stackrel{?}{=} J_M + H(J_M \|\| SK_{SMV} \|\|$	$SK_{MVS} \|\| TS_M \|\| Sig_M)$
$TID_{SND} \|\| TS_M) \cdot Pub_M$. If so, compute	$\overleftarrow{Msg_{SM_2} : \langle J_M, Sig_M, TID_M, RID_M, TID_{SND}^*, TS_M \rangle}$
$TID_{SND}^{new} = TID_{SND}^* \oplus$	
$H(TID_{SND} \|\| SK_{SMV} \|\| TS_M \|\| Sig_M)$.	
Update TID_{SND} with TID_{SND}^{new}.	
Generate $TID_M^{new} \in Z_q^*$, timestamp TS_{SM}.	
Compute	
$TID_M^* = TID_M^{new} \oplus H(TID_M \|\| SK_{SMV} \|\| TS_{SM})$,	Check if $\|TS_M^* - TS_M\| \leq \Delta$T? If so, compute
$SKV_{SMV} = H(SK_{SMV} \|\| TS_{SM} \|\| TID_M^{new})$.	$TID_M^{new} = TID_M^* \oplus H(TID_M \|\| SK_{MVS} \|\| TS_{SM})$,
$\underrightarrow{Msg_{SM_3} : \langle TID_M^*, SKV_{SMV}, TS_{SM} \rangle}$	$SKV_{MVS} = H(SK_{MVS} \|\| TS_{SM} \|\| TID_M^{new})$.
	If $SKV_{SMV} \stackrel{?}{=} SKV_{MVS}$, store SK_{MVS}.
Store SK_{SMV}.	Update TID_M with TID_M^{new}.

Figure 7.6: Summary of authentication phase between SN and MV

- **Step** $MVFS_1$**:** MV requests the fog server FS for the transaction containing TID_M from the blockchain. FS retrieves and sends $Tx_i = \langle\ TID_M,\ K_M^*,\ E_{Pub_M}(ID_M, K_{MV_i,FS_j}, RID_M, TS_{mc})\rangle$ to the MV.

- **Step** $MVFS_2$**:** MV decrypts TX_i using the private key pr_M to obtain $\langle ID_M, K_{MV_i,FS_j}, RID_M, TS_{mc}\rangle$. It then generates a private random secret $u_M \in Z_q^*$ and a timestamp TS_{mf1}, and hides RID_M as $RID_M^* = RID_M \oplus H(K_{MV_i,FS_j} \|\| TS_{mf1})$ and computes $U_M = H(u_M \|\| pr_M \|\| TID_M \|\| TS_{mf1}) \cdot G$. TRA's contribution to the session key for MV is extracted from K_M^* as $H(K_M \|\| TS_{mc}) = K_M^* \oplus H(K_{MV_i,FS_j} \|\| TID_M \|\| RID_M \|\| TS_{mc})$ using association key K_{MV_i,FS_j} and TID_M, RID_M, and TS_{mc}. It is hidden as $K_M^h = H(K_M \|\| TS_{mc}) \oplus$

Mobile Vehicle(MV)	Fog Server(FS)
$\xrightarrow{\langle AuthCredBlock_{request}, TID_M \rangle}$	Retrieve $Tx_i = \langle\ TID_M,\ K_M^*,\ E_{Pub_M}(ID_M,$ $K_{MV_i,FS_j}, RID_M, TS_{mc})\ \rangle$ using TID_M.
	$\xleftarrow{\langle AuthCredBlock_{response}, Tx_i \rangle}$
Compute $\langle ID_M, K_{MV_i,FS_j}, RID_M, TS_{mc} \rangle$ $= Dec_{pr_M}(Enc_{Pub_M}(ID_M,\ K_{MV_i,FS_j}, RID_M, TS_{mc}))$.	
Generate $u_M \in Z_q^*$, timestamp TS_{mf1}.	
Compute $RID_M^* = RID_M \oplus H(K_{MV_i,FS_j} \|\| TS_{mf1})$,	
$U_M = H(u_M \|\| pr_M \|\| TID_M \|\| TS_{mf1}) \cdot G$,	Check if $\|TS_{mf1}^* - TS_{mf1}\| \leq \Delta$ T? If so,
$H(K_M \|\| TS_{mc}) = K_M^* \oplus H(K_{MV_i,FS_j} \|\|$	compute $RID_M = RID_M^* \oplus H(K_{MV_i,FS_j} \|\| TS_{mf1})$,
$TID_M \|\| RID_M \| \| TS_{mc})$,	$H(K_M \|\| TS_{mc}) = K_M^h \oplus H(RID_M \|\| TID_M \|\|$
$K_M^h = H(K_M \|\| TS_{mc}) \oplus H(RID_M \|\|$	$Pub_F \|\| K_{MV_i,FS_j} \|\| TS_{mf1})$.
$TID_M \|\| Pub_F \|\| K_{MV_i,FS_j} \|\| TS_{mf1})$,	Verify $Sig_M \cdot G \stackrel{?}{=} U_M + H(TID_M \|\| RID_M \|\| Pub_M \|\|$
$Sig_M = H(u_M \|\| pr_M \|\| TID_M \|\| TS_{mf1}) + H(TID_M \|\|$	$Pub_F \|\| H(K_M \|\| TS_{mc}) \ \|\ TS_{mf1}) \cdot Pub_M$.
$RID_M \|\| Pub_M \|\| Pub_F \|\| H(K_M \|\| TS_{mc}) \ \|\ TS_{mf1})$	If valid, generate timestamp TS_{mf2}.
$^* pr_M \pmod q$.	Retrieve $Tx_j = \langle\ TID_F,\ K_F^*,$
$\xrightarrow{Msg_{MF_1} : \langle U_M, TID_M, RID_M^*, Sig_M, K_M^h, TS_{mf1} \rangle}$	$E_{Pub_F}(ID_F, K_{MV_i,FS_j},\ RID_F, TS_{fc})\rangle$ using TID_F.
	Compute $\langle\ ID_F, K_{MV_i,FS_j}, RID_F, TS_{fc}\ \rangle$ $= Dec_{pr_F}(Enc_{Pub_F}(ID_F,\ K_{MV_i,FS_j},\ RID_F, TS_{fc})\ \rangle$,
	$RID_F^* = RID_F \oplus H(K_{MV_i,FS_j} \|\| TS_{mf2})$,
	$H(K_F \| \| TS_{fc}) = K_F^* \oplus H(K_{MV_i,FS_j} \|\| TID_F \|\|$
	$RID_F \|\| TS_{fc})$,
	$K_F^h = H(K_F \| \| TS_{fc}) \oplus H(RID_F \|\| TID_F \|\|$
	$Pub_M \|\| K_{MV_i,FS_j} \|\| TS_{mf2})$.
	Generate $v_F \in Z_q^*$.
	Compute $V_F = H(v_F \|\| pr_F \|\| TID_F \|\| TS_{mf2}) \cdot G$,
	$DK_{FM} = H(v_F \|\| pr_F \|\| TID_F \|\| TS_{mf2}) \cdot U_M$,
Check if $\|TS_{mf2}^* - TS_{mf2}\| \leq \Delta T$? If so, compute	$SK_{FM} = H(H(K_M \|\| TS_{mc}) \|\| U_M \|\| V_F \|\|$
$RID_F = RID_F^* \oplus H(K_{MV_i,FS_j} \|\| TS_{mf2})$,	$H(K_F \|\| TS_{fc}) \ \|\ DK_{FM})$,
$H(K_F \| \| TS_{fc}) = K_F^h \oplus H(RID_F \|\| TID_F \|\|$	$Sig_F = H(v_F \|\| pr_F \|\| TID_F \|\| TS_{mf2}) + H(SK_{FM} \|\|$
$Pub_M \|\| K_{MV_i,FS_j} \|\| TS_{mf2})$,	$DK_{FM} \|\| RID_M \|\| RID_F \|\| H(K_F \|\|$
$DK_{MF} = H(u_M \|\| pr_M \|\| TID_M \|\| TS_{mf1}) \cdot V_F$,	$TS_{fc}) \|\| TS_{mf2}) \ ^* pr_F \pmod q$.
$SK_{MF} = H(H(K_M \|\| TS_{mc}) \|\| U_M \|\|$	Generate $TID_M^{new} \in Z_q^*$. Compute $TID_M^* = TID_M^{new}$
$V_F \ \|\ H(K_F \| \| TS_{fc}) \|\| DK_{MF})$.	$\oplus\ H(TID_F \|\| RID_F \|\| U_M \|\| TID_M \|\| RID_M \|\|$
Verify $Sig_F \cdot G \stackrel{?}{=} V_F + H(SK_{MF} \|\| DK_{MF} \|\| RID_M \|\|$	$H(K_F \|\| TS_{fc}) \|\| H(K_M \|\| TS_{mc}))$.
$RID_F \|\| H(K_F \| \| TS_{fc}) \ \|\ TS_{mf2}) \cdot Pub_F$. If so,	
compute $TID_M^{new} = TID_M^* \oplus H(TID_F \|\| RID_F \|\| U_M$	$Msg_{MF_2} : \langle V_F, Sig_F, TID_M^*,$
$\| \| TID_M \|\| RID_M \|\| H(K_F \| \| TS_{fc}) \ \|\ H(K_M \|\| TS_{mc}))$.	$\xleftarrow{TID_F, RID_F^*, K_F^h, TS_{mf2} \rangle}$
Update TID_M with TID_M^{new} in its database.	
Generate $TID_F^{new} \in Z_q^*$ and compute	
$TID_F^* = TID_F^{new} \oplus H(TID_M \|\| RID_M \|\| V_F \|\|$	Check if $\|TS_{mf3}^* - TS_{mf3}\| \leq \Delta$T? If so, compute
$TID_F \|\| RID_F \|\| H(K_M \|\| TS_{mc}) \|\| H(K_F \| \| TS_{fc}))$.	$TID_F^{new} = TID_F^* \oplus H(TID_M \|\| RID_M \|\| V_F \|\|$
Generate TS_{mf3} and compute	$TID_F \|\| RID_F \|\| H(K_M \|\| TS_{mc}) \|\| H(K_F \| \| TS_{fc}))$
$SKV_{mf} = H(SK_{mf} \|\| TS_{mf3} \|\| TID_F^{new})$.	$SKV_{fm} = H(SK_{fm} \|\| TS_{mf3} \|\| TID_F^{new})$
$\xrightarrow{Msg_{MF_3} : \langle SKV_{mf}, TS_{mf3}, TID_F^* \rangle}$	Verify $SKV_{fm} \stackrel{?}{=} SKV_{mf}$. If so,
	store SK_{fm} in memory as the session key.
Store SK_{mf} in memory as the session key.	and update TID_F with TID_F^{new} in its database.

Figure 7.7: Summary of authentication phase between MV and FS

$H(RID_M||\ TID_M||\ Pub_F||\ K_{MV_i,FS_j}||\ TS_{mf1})$. It computes a signature over u_M as $Sig_M = H(u_M||\ pr_M||\ TID_M||\ TS_{mf1}) + H(TID_M||\ RID_M||\ Pub_M||\ Pub_F||\ H(K_M||\ TS_{mc})||\ TS_{mf1})\ *\ pr_M \pmod q$. MV sends the message $Msg_{MF_1} = \langle U_M, TID_M, RID_M^*, Sig_M, K_M^h, TS_{mf1}\rangle$ to FS.

- **Step** $MVFS_3$**:** FS verifies the timestamp of the message received at TS_{mf1}^* as $|TS_{mf1}^* - TS_{mf1}| \leq \Delta$ T. It extracts RID_M from RID_M^* using association key K_{MV_i,FS_j} as $RID_M = RID_M^* \oplus H(K_{MV_i,FS_j}||\ TS_{mf1})$ and TRA's session key contribution to MV as $H(K_M||\ TS_{mc})$ from K_M^h using association key K_{MV_i,FS_j} and other received parameters RID_M, TID_M, TS_{mf1} and publicly available Pub_F. FS extracts $H(K_M||\ TS_{mc}) = K_M^h \oplus H(RID_M||\ TID_M||\ Pub_F||\ K_{MV_i,FS_j}||\ TS_{mf1})$ and verifies the signature as $Sig_M \cdot G \stackrel{?}{=} U_M + H(TID_M||\ RID_M||\ Pub_M||\ Pub_F||\ H(K_M||\ TS_{mc})||\ TS_{mf1}) \cdot Pub_M$.
- **Step** $MVFS_4$**:** FS generates a timestamp TS_{mf2} if Sig_M is verified to be correct. It retrieves the block with transaction containing TID_F as $Tx_j = \langle TID_F,\ K_F^*,\ E_{Pub_F}(ID_F, K_{MV_i,FS_j}, RID_F, TS_{fc})\rangle$. It uses its own private key pr_F to decrypt and obtain $\langle ID_F,\ K_{MV_i,FS_j},\ RID_F,\ TS_{fc}\rangle$. The extracted RID_F is hidden as $RID_F^* = RID_F \oplus H(K_{MV_i,FS_j}||\ TS_{mf2})$. $TRA's$ session key contribution for FS is extracted from K_F^* as $H(K_F||TS_{fc})$ from K_F^* using association key K_{MV_i,FS_j} and other received parameters TID_F, RID_F and TS_{fc} as $K_F^* = K_F^* \oplus H(K_{MV_i,FS_j}||\ TID_F||\ RID_F||\ TS_{fc})$ and hidden as $K_F^h = H(K_F||TS_{fc}) \oplus H(RID_F||\ TID_F||\ Pub_M||\ K_{MV_i,FS_j}||\ TS_{mf2})$.
- **Step** $MVFS_5$**:** FS generates $v_F \in Z_q^*$ and computes $V_F = H(v_F||\ pr_F||\ TID_F||\ TS_{mf2}) \cdot G$, the Diffie-Hellman type key as $DK_{FM} = H(v_F||\ pr_F||\ TID_F||\ TS_{mf2}) \cdot U_M$ and finally the session key with MV as $SK_{FM} = H(H(K_M||\ TS_{mc})||\ U_M||\ V_F||\ H(K_F||TS_{fc})||\ DK_{FM})$. SK_{FM} consists of five parts, a) TRA's contribution for MV, $H(K_M||\ TS_{mc})$, b) MV's contribution, $H(u_M||\ pr_M||\ TID_M||\ TS_{mf1})$ c) FS's contribution, $H(v_F||\ pr_F||\ TID_F||\ TS_{mf2})$ d) TRA's contribution for FS, $H(K_F||TS_{fc})$ and e) Diffie-Hellman type parameter, DK_{FM}. A signature is generated over v_F as $Sig_F = H(v_F||\ pr_F||\ TID_F||\ TS_{mf2}) + H(SK_{FM}||\ DK_{FM}||\ RID_M||\ RID_F||\ H(K_F||TS_{fc})\ ||\ TS_{mf2})\ *\ pr_F \pmod q$. MV generates a new temporary identity as $TID_M^{new} \in Z_q^*$ and hides it as $TID_M^* = TID_M^{new} \oplus H(TID_F||\ RID_F||\ U_M||\ TID_M||\ RID_M||\ H(K_F||TS_{fc})\ ||\ H(K_M||\ TS_{mc}))$ to be sent to MV in the message The message $Msg_{MF_2} : \langle V_F, Sig_F, TID_M^*, TID_F, RID_F^*, K_F^h, TS_{mf2}\rangle$ is then sent to the MV.

- **Step** $MVFS_6$**:** MV receives Msg_{MF_2} at TS^*_{mf2} and verifies the timestamp as $|TS^*_{mf2} - TS_{mf2}| \leq \Delta\text{T}$. It extracts $RID_F = RID^*_F \oplus H(K_{MV_i,FS_j}||\ TS_{mf2})$ and. TRA's contribution for FS is extracted as $H(K_F||TS_{fc}) = K^h_F \oplus H(RID_F||\ TID_F||\ Pub_M||\ K_{MV_i,FS_j}||\ TS_{mf2})$. It then computes the Diffie-Hellman parameter $DK_{MF} = H(u_M||\ pr_M||\ TID_M||\ TS_{mf1}) \cdot V_F$ and the session key as $SK_{MF} = H(H(K_M||\ TS_{mc})||\ U_M||\ V_F||\ H(K_F||TS_{fc})||\ DK_{MF})$. It then verifies FS's signature on Msg_{MF_2} as $Sig_F \cdot G \stackrel{?}{=} V_F + H(SK_{FM}||\ DK_{MF}||\ RID_M||\ RID_F||\ H(K_F||TS_{fc})||\ TS_{mf2}) \cdot Pub_F$. If so, it extracts TID^{new}_M from $TID^*_M = TID^*_M \oplus H(TID_F||\ RID_F||\ U_M||\ TID_M||\ RID_M||\ H(K_F||TS_{fc})\ ||\ H(K_M||\ TS_{mc}))$ and updates TID_M with TID^{new}_M in its database.
- **Step** $MVFS_7$**:** MV creates a new temporary session identity for FS as $TID^{new}_F \in Z^*_q$ and hides it as $TID^*_F = TID^{new}_F \oplus H(TID_M||\ RID_M||\ V_F||\ TID_F||\ RID_F||\ H(K_M||\ TS_{mc})\ ||\ H(K_F||TS_{fc}))$. It then generates a new timestamp TS_{mf3} and computes a session key verifier $SKV_{mf} = H(SK_{mf}||\ TS_{mf}||\ TID^{new}_F)$. The final message for this phase is sent from MV to FS as $Msg_{MF_3} : \langle SKV_{mf}, TS_{mf3}, TID^*_F\rangle$.
- **Step** $MVFS_8$**:** FS receives Msg_{MF_3} at time TS^*_{mf3}, verifies the timestamp by the condition: $|TS^*_{mf3} - TS_{mf3}| \leq \Delta\text{T}$ and extracts TID^{new}_F from TID^*_F as $TID^{new}_F = TID^*_F \oplus H(TID_M||\ RID_M||\ V_F||\ TID_F||\ RID_F||\ H(K_M||\ TS_{mc})\ ||\ H(K_F||TS_{fc}))$. It then computes the session key verifier as $SKV_{fm} = H(SK_{fm}||\ TS_{mf3}||\ TID^{new}_F)$. If the computed SKV_{fm} and received SKV_{mf} match, then it stores SK_{fm} in memory and updates TID_F with TID^{new}_F in its database. MV then stores SK_{mf} in memory as the session key for further communication exchange between MV and FS.

The summary of the MVFS authentication phase is provided in Figure 7.7.

7.4.5 Secure data aggregation with block creation, verification and addition in blockchain

This section provides a compendious presentation of the creation of transactions by the TRA and the creation of blocks by the fog server FS with the following steps.

- **Step** $BCFS_1$**:** The TRA creates transactions $Tx_i = \langle TID_M,\ K^*_M,\ E_{Pub_M}(ID_M,\ K_{MV_i,FS_j},\ RID_M,\ TS_{mc})\rangle$ and $Tx_j = \langle TID_F,\ K^*_F,\ E_{Pub_F}(ID_F,\ K_{MV_i,FS_j},\ RID_F,\ TS_{fc})\rangle$ for the mobile vehicle MV and the fog server FS, respectively, during

registration as shown in Section 2 and 3, respectively. TRA signs the transactions with the ECDSA signature generation algorithm $sig(\cdot)$ using pr_{TRA} as $Sig_{Tx_i} = ECDSA.sig_{pr_{TRA}}(Tx_i)$ and $Sig_{Tx_j} = ECDSA.sig_{pr_{TRA}}(Tx_j)$. TRA sends $\langle Tx_i, Sig_{Tx_i} \rangle$ to the FS associated with the respective MV and also sends $\langle Tx_j, Sig_{Tx_j} \rangle$ to the registering fog server FS. It is to be noted that only part of each transaction is encrypted. This suggests the use of consortium (hybrid) blockchain for the purpose of our scheme.

- **Step** $BCFS_2$**:** The fog server FS verifies the signatures Sig_{Tx_i} and Sig_{Tx_j} for Tx_i and Tx_j, respectively. If the signatures are valid, the FS marks them as valid. After the FS receives n_m transactions from the TRA, it creates a block $Block_m$ with a message $BlockMsg_m = (Tx_{i1} \,||Tx_{i2}\, || \cdots \,||Tx_{in_m})$ and a signature on the block as $Sig_{pr_F}(BlockMsg_m)$. Similarly, after the FS receives n_f transactions from the TRA, it creates a block $Block_f$ with the message as $BlockMsg_f = (Tx_{j1} \,||Tx_{j2}\, || \cdots ||Tx_{jn_f})$ and a signature on the block as $Sig_{pr_F}(BlockMsg_f)$. Such created blocks have the formats as shown in Figure 7.8 with the BlockType designated as "$AuthCred_m$" and "$AuthCred_f$" for mobile vehicles and fog servers, respectively. Once a block is created by the fog server, the following tasks are performed: a) the fog system executes a leader selection algorithm to select a fog server as the leader using [348], and b) a consensus algorithm is executed to validate the block and add the block to the blockchain using Algorithm 4.

- **Step** $BCFS_3$**:** During the $MVFS$ phase of the authentication scheme as shown in Section 2, the blockchain is searched for the block containing the transaction with the required TID_M or TID_F, and the AuthCred block is retrieved from the chain and sent to the requesting entity. Based on this retrieved block from the chain, the rest of the steps in the $MVFS$ phase proceeds. If the scheme succeeds in executing all its steps, a session key is established between MV and FS.

- **Step** $BCFS_4$**:** The mobile vehicle MV collects sensitive sensor data $SNDATA$ from the IoT smart sensor devices SN corresponding to its associated zone in the farm field. This data is encrypted by SN using the session key SK_{SMV} established in $SNMV$ phase as shown in Section 1. The encrypted data is sent to the mobile vehicle MV_i. The mobile vehicle MV_i decrypts the data with SK_{MVS}. MV_i then encrypts $SNDATA$ with SK_{MF} and sends it to the fog server FS_j, which decrypts using SK_{FM}.

- **Step** $BCFS_5$**:** The fog server FS_j creates a transaction $Tx_d = \langle ID_F, TS_F, Z_i, SNDATA \rangle$ for each received sensor data $SNDATA$ and creates a "Sensor-Data" block $Block_d$ as shown in Figure 7.9 with the n_d collected transactions from $BlockMsg_d = (Enc_{Pub_F}(Tx_{d1}) || Enc_{Pub_F}(Tx_{d2}) || \cdots || Enc_{Pub_F}(Tx_{dn_d}))$ and a signature on the block as $Sig_{pr_F}(BlockMsg_d)$. The BlockType is designated as "Sensor-Data". After the creation of the block, the fog servers in the fog system select a leader and execute the consensus algorithm provided in Algorithm 4 to validate the block and add it into the blockchain.

To analyse the storage space and communication cost, the following notations are used. The identities and random secrets are considered to be of 160 bits each. The length of output of hash function, ciphertext block of "symmetric key encryption/decryption using the Advanced Encryption Standard (AES-128) encryption algorithm" are taken as 256 bits and 128 bits, respectively. For public key cryptographic operations, the "Elliptic Curve Cryptosystem (ECC)" is chosen such that 160-bit ECC provides the same security level as that for 1024-bit RSA cryptosystem [66]. An elliptic curve point $A = (x_A, y_A)$ needs $(160 + 160) = 320$ bits, and the timestamp requires 32 bits. To analyse the computational operations, the following notations are also used. T_h stands for the hash operation performed using SHA-256 hash algorithm, T_{ecm} and T_{eca} represent the elliptic curve multiplication and addition, respectively. T_{senc} and T_{sdec} denote the AES-128 symmetric encryption/decryption operations, respectively. T_{CKDf} represents the cryptographic key derivation function, which is considered as T_{ecm}. The time for signature generation using $ECDSA$ is $T_h + T_{ecm}$ while the verification time using $ECDSA$ takes $2T_{ecm} + T_{eca} + T_h$. The time T_{ECCEnc} for ECC encryption takes $2T_{ecm} + T_{eca}$, while the time T_{ECCDec} for ECC decryption needs $T_{ecm} + T_{eca}$ operations. Additionally, T_{mul} and T_{add} represent "modular multiplication" and "modular addition" over the finite field $GF(q)$, respectively.

The size of AuthCred block as shown in Figure 7.8 is computed with the sizes of the components $\{BVer_{ac}, PBH_{ac}, BlockType, MTR_{ac}, TS_{ac}, OWN_{ac}, Pub_{TRA}, Pub_F, \{(Tx_i, ECDSA.sig_{Tx_i})\}_{i=1,2,\cdots,n_m}, Sig_{pr_F}(BlockMsg_m), CBHash\}$ as 32, 256, 32, 256, 32, 160, 320, 320, $1472 * n_m$, 320, and 256 bits, respectively. Therefore, the total size of AuthCred block becomes $1984 + 1472n_m$ bits. Similarly, the size of AuthCred block for the FS point of view shown in Figure 7.8 is computed with the sizes of $\{BVer_{ac}, PBH_{ac}, BlockType, MTR_{ac}, TS_{ac}, OWN_{ac}, Pub_{TRA}, Pub_F, \{(Tx_j, ECDSA.sig_{Tx_j})\}_{j=1,2,\cdots,n_f}$,

$Sig_{pr_F}(BlockMsg_f)$, $CBHash\}$ as 32, 256, 32, 256, 32, 160, 320, 320, $1472 * n_f$, 320, and 256 bits, respectively. Therefore, the total size of the AuthCred block becomes 1984 $+1472n_f$ bits for blocks with transactions from FS. The size of SensorData block in Figure 7.9 is computed with the sizes of $\{BVer_d$, PBH_d, $BlockType$, MTR_d, TS_d, OWN_d, Pub_F, $\{(E_{Pub_F}(Tx_d)$, $ECDSA.sig_{Tx_d})\}_{d=1,2,\cdots,n_d}$, $Sig_{pr_F}(BlockMsg_d)$, $CBHash\}$ as 32, 256, 32, 256, 32, 160, 320, $640 * n_d$, 320, and 256 bits, respectively, and the total size becomes 1664 $+640n_d$ bits.

Block Header	
Block Version ($BVer_{ac}$)	Unique serial number
Previous Block Hash (PBH_{ac})	Hash value of previous block
BlockType	Authentication Credentials ($AuthCred_m$ or $AuthCred_f$)
Merkle Tree Root (MTR_{ac})	Merkle tree root on transactions
Timestamp (TS_{ac})	Block creation time
Owner of Block OWN_{ac}	Fog server (FS_j)
Public key of transactions verification	Pub_{CRA}
Public key of block signer	Pub_F
Block Payload (Transactions)	
MV Transactions Tx_i OR FS Transactions Tx_j	$\{(Tx_{il}, ECDSA.sig_{Tx_{il}})\|$ $l = 1, 2, \cdots, n_m\}$ $\vdots$ $\{(Tx_{jl}, ECDSA.sig_{Tx_{jl}})\|$ $l = 1, 2, \cdots, n_f\}$
ECDSA signature on Block	$Sig_{pr_F}(BlockMsg_m)$ or $Sig_{pr_F}(BlockMsg_f)$
Current Block Hash ($CBHash_{ac}$)	Hash value of present block

Figure 7.8: Structure of an AuthCred block $Block_m$ or $Block_f$ in the blockchain

The proposed consensus Algorithm 4 is vote-based with threshold in "Practical Byzantine Fault Tolerance (PBFT)" as described in [95]. A leader fog node broadcasts a block $Block_k$ to the follower fog nodes and collects their vote on its validity for addition to the blockchain. Every fog server validates the block hash, block signature, and Merkle tree root and prepares its vote. The fog server encrypts its vote with the leader's public key using ECC encryption before sending the response in the public channel. The leader

Block Header	
Block Version	$BVer_d$
Previous Block Hash	PBH_d
BlockType	Sensing Data ($SensorData$)
Merkle Tree Root	MTR_d
Timestamp	TS_d
Owner of Block OWN_d	FS_j
Public Key of Signer FS	Pub_F
Block Payload (Encrypted Transactions)	
List of n_d Encrypted Transactions $\#l$ (Tx_{dl})	$\{(E_{Pub_F}(Tx_{dl}), ECDSA.sig_{Tx_{dl}}) \mid l = 1, 2, \cdots, n_d\}$
ECDSA Signature on Block	$Sig_{pr_F}(BlockMsg_d)$
Current Block Hash	$CBHash_d$

Figure 7.9: Structure of a SensorData block $Block_d$

decrypts each vote using ECC decryption algorithm. If the vote favors block addition, the leader increments the favorable vote count. The validated block is added to the chain when the total positive votes are greater than the fault limit.

1) **Computational complexity**: Any step in Algorithm 4 that is directly linked to a change in the blockchain is considered on-chain. If a block fails verification or does not receive the required votes, it will not be added to the chain and thus, it cannot affect the blockchain state.

   • **Off-chain**: Steps 1 to 26 in the Algorithm 4 are executed off-chain. The verification of the block hash requires T_h operations. The Merkle tree root contains N_k nodes and takes $O(\log_2 N_k)$ operations. Thus, the total computational complexity for N_f fog nodes is at least N_f * $(2T_h + 5T_{ecm} + 3T_{eca} + O(\log_2 N_k))$.

   • **On-chain**: Steps 27 and 28 in the Algorithm 4 are executed on-chain. Adding a block to the blockchain consists of finding the most recent block and linking the new block to the chain. As these steps take constant time, the on-chain computational complexity is constant.

   The total computational complexity is denoted and estimated by $C_{comp} = N_f$ * $(2T_h + 5T_{ecm} + 3T_{eca} + O(\log_2 N_k)) + O(1)$.

2) **Communication complexity**: The size of AuthCred block is $(1984 + 1472n_m)$ and

Algorithm 4 Achievement of consensus of the blockchain

Input: $Block_k$: A full block with the structure as given in Figure 7.8 or Figure 7.9 that is to be added to the blockchain, N_f: Total number of P2P nodes (fog servers) in the blockchain network where $k = m$ for $Block_m$ or $k = f$ for $Block_f$ or $k = d$ for $Block_d$, $Block_k$ has n_m or n_f or n_d transactions according to the block type being added

Output: Status of block commit operation (YES/NO)

1: Select a leader FS_L among the fog nodes
2: Set $Limit_{Votes} = 2 * N_{faulty} + 1$
3: $AllFogVotes \leftarrow$ NIL
4: FS_L broadcasts $Block_k$ to all fog nodes in a Peer-to-Peer (P2P) network
5: **for** each fog server node FS_j in the P2P network **do**
6: Set $Fog\_Vote_j = NO$
7: Compute $Block\_Hash = H(Block_k)$
8: **if** ($Block\_Hash = CBHash$) **then**
9: **if** (block signature is valid) **then**
10: Create Merkle tree root (MTR_{Block}) with n_k transactions from block payload
11: **if** (MTR_{Block} matches with the MTR in the block) **then**
12: Set $Fog\_Vote_j = YES$
13: FS_j encrypts Fog_Vote_j using the public key Pub_{F_L} of FS_L as $ECCEnc_{Pub_{F_L}}(Fog\_Vote_j)$
14: Add $ECCEnc_{Pub_{F_L}}(Fog\_Vote_j)$ to $AllFogVotes$ and send it to FS_L
15: Set $Vote_{Count} \leftarrow 0$
16: **for** each encrypted vote V_t reply in $AllFogVotes$ **do**
17: FS_l computes $V_t = ECCDec_{pr_{F_L}}(ECCEnc_{Pub_{F_L}}(V_t))$, where $V_t = Fog\_Vote_j$
18: **if** (V_t is valid) **then**
19: Set $Vote_{Count} = Vote_{Count} + 1$
20: **if** ($Vote_{Count} \geq Limit_{Votes}$) **then**
21: FS_l adds block $Block_k$ into its blockchain
22: FS_L broadcasts block commit status as YES to the blockchain network
23: Other peer fog nodes add the block into their blockchains

SensorData block is (1664 +640n_d) bits. Note that 160*N_f bits are communicated for the encrypted votes from N_f fog servers. Thus, The algorithm uses N_f * (1984 +1472n_m) + N_f (160) bits for communicating the AuthCred block and N_f * (1664

$+640n_d$) + N_f (160) bits to communicate SensorData block.

3) **Fault tolerance**: Let N_{fault} be the number of faulty fog servers in a fog system with N_f servers. A consensus is reached even if only N_f - N_{fault} servers communicate among themselves. However, the unresponsive N_{fault} fog servers might not be faulty, and the responsive N_{fault} fog servers might be faulty. Even with these cases a consensus is attainable if responses from the non-faulty fog servers outnumber the responses from the faulty ones under the condition N_f - $2N_{fault} > N_{fault}$ i.e $N_f > 3N_{fault}$. The fog system achieves a consensus if the system has at least $N_f = 3N_{fault} + 1$ fog servers. Out of these, when $2N_{fault} + 1$ servers reach a consensus, the fog system reaches a consensus too. In other words, out of any number of N_f servers, if $1/3^{rd}$ nodes are faulty, the consensus is declared for positive responses from $2/3^{rd}$ servers, that is, when positive votes are $2(N_f - 1)/3 + 1$.

4) **Blockchain retrieval time**: Let len_{BC} be the length of the blockchain at a time when the Big Data Centre requests data from the blockchain. The blocks are traversed using the previous block hash, which requires T_h time to compute the block hash. Thus, the time to access the blockchain is given by T_h * len_{BC}.

5) **Latency and energy consumption**: The proposed consensus algorithm does not solve any hashing puzzles. Instead, similar to PBFT, it uses regular communication to reach a consensus. It is highly efficient for small-scale networks. Let L_{block} be the transmission latency of broadcasting a new block to all nodes, ET be the effective throughput, L_{Vote} be the transmission latency of vote responses, L_V be the latency of block verification, $size_{block}$ is the block size, $size_{Vote}$ be the size of a vote response message. Then the transmission latency of block broadcast and vote responses are given as $L_{block} = size_{block}/ET$ * N_f^2 and $L_{Vote} = size_{Vote}/ET$ * N_f^2, respectively. If P_{Tr} is the transmission power, then the energy consumption for transmission is given as P_{Tr} * $(L_{block} + L_{Vote})$ * $N_f^2 = P_{Tr}$ * $(size_{block}/ET + size_{Vote}/ET)$ * N_f^2. If P_{Comp} is the computational power, then the energy consumption for computation is given as P_{Comp} * L_V, where L_V is given as $size_{block}/Cap_{CPU}$, with C_{CPU} being the CPU capacity of the fog nodes. Thus, the total energy consumption of one new block with the proposed consensus Algorithm 4 is given as P_{Tr} * N_f^2 * $(size_{block}/ET + size_{Vote}/ET) + P_{Comp}$ * N_f * $(size_{block}/Cap_{CPU})$ [168].

Remark 7.1. *The issue of storing a huge amount of data on the blockchain and its effect on retrieval time is related to the problem of implementation of the hybrid blockchain in*

the proposed scheme. Applying efficient data structures to store the blockchain can lead to efficient retrievals. Merkle tree can be implemented as B-Merkle tree using modified polynomial commitment scheme with proofs based on element ordering giving small proof sizes and low tree heights [291, 292]. Tu et al. [316] suggests B+ tree to store the blockchain keeping the transactions on leaf nodes and the indexes on inner nodes. Redis Cache technology is used for fast indexing of block files. Their proposed retrieval algorithm has better retrieval efficiency. Feng et al. [135] proposed the more efficient BB+ tree with Bloom filter for large blocks which improves the retrieval time by roughly 40% and 43% using single and multiple features, respectively. Thus, usage of such efficient methods to store and retrieve from the blockchain will ensure that the access of the hybrid blockchain during authentication does not slow down the overall performance of the proposed scheme.

7.4.6 Dynamic nodes addition phase

This phase is used whenever an IoT smart device is corrupted or damaged and needs to be replaced with a new node. The TRA registers the newly placed node, and assigns the appropriate credentials consisting of identities, and public-private key pairs required for authentication.

- **Step** DNA_1**:** TRA picks ID_{SND}^{new}, $s^{new} \in Z_q^*$, a registration timestamp RTS_S^{new} and computes the pseudo-identity as $RID_{SND}^{new} = H(ID_{SND}^{new} || \ s^{new} || \ RTS_S^{new} || \ pr_{TRA})$ and the temporary identity as $TID_{SND}^{new} = H(RID_{SND}^{new} || \ s^{new} || \ pr_{TRA} || \ RTS_S^{new})$. TRA picks the private key as $pr_S^{new} \in Z_q^*$ and the corresponding public key as $Pub_S^{new} = pr_S^{new} \cdot G$.
- **Step** DNA_2**:** TRA pre-loads SN with $\{(RID_{SND}^{new}, \ TID_{SND}^{new}), \ H(\cdot), \ E_q(\kappa, \mu), \ G, \ (pr_S^{new}, \ Pub_S^{new})\}$.

7.5 Security analysis

In this section, the security strength of the proposed scheme is analyzed using the "Real-or-Random (ROR) model" [43] and "Random Oracle Model (ROM)" [218] in formal security analysis. In addition, the proposed scheme is proved to be insusceptible to many widely known attacks with a non-mathematical (informal) security analysis. The scheme is then simulated for formal security verification under the widely recognized "Automated

Validation of Internet Security Protocols and Applications (AVISPA)" software validation tool [9] to verify its safety.

7.5.1 Formal security analysis under ROR model

In the proposed scheme, the IoT smart device SN and the mobile vehicle MV that collects data from SN establish a session key $SK_{SMV} = (SK_{MVS})$ during the SNMV phase (see Section 1). The mobile vehicle MV and the fog server FS to which the data is deposited establish another session key $SK_{MF} = (SK_{FM})$ during the $MVFS$ phase (see Section 2).

1) **Random oracle model**: We define the respective security model based on the works by Bellare *et al.* [70] and Wu *et al.* [341], for the proposed scheme through a sequence of the interactive games between a challenger and an adversary. We prove that the proposed scheme provides the session key security against the adversary. For this purpose, the security model to analyze $AgroMobiBlock$ is defined as follows.

   - **Random oracle**: A "one-way cryptographic hash function" $H(\cdot)$ that acts as a random oracle, say $Hash$ is accessible to all the entities.
   - **Participants**: Let $\Psi_{SN}^{i_1}$, $\Psi_{MV}^{i_2}$ and $\Psi_{FS}^{i_3}$ denote the i_1^{th}, i_2^{nd} and i_3^{rd} instances of the participants SN, MV and FS, respectively, which are known as "random oracles".
   - **Freshness of instance**: An instance Ψ^i is fresh if it satisfies the conditions: 1)Ψ^i and its partner share a session key. 2) Neither Ψ^i nor its partner has received the reveal query, implying that the session key is not accessible to adversary $\mathcal{A}$.
   - **Freshness of session key**: A session key is deemed fresh if it satisfies the conditions: 1)The instances Ψ^{i_1} and Ψ^{i_2} are partners. 2) The instances Ψ^{i_1} and Ψ^{i_2} are fresh. 3) The instances Ψ^{i_1} and Ψ^{i_2} share a session key.
   - **Accepted state**: An instance Ψ^i initially in the waiting state moves to the accepted state when it receives the expected message.
   - **Adversary**: An adversary $\mathcal{A}$ can avail the queries shown in Table 7.2.

Definition 7.1 (Semantic security)**.** *Let* $Adv_{\mathcal{A}}^{AgroMobiBlock}(t_p)$ *denote the "advantage of an adversary* $\mathcal{A}$*, running in polynomial time* t_p *in breaking the semantic security of the proposed scheme* $AgroMobiBlock$ *in order to derive the session key* $SK_{SMV} = SK_{MVS}$ *between the IoT smart device* SN *and the mobile vehicle* MV *in the SNMV*

Table 7.2: Queries and their purposes

Query	Purpose
$Execute(\Psi_{SN}^{i_1}, \Psi_{MV}^{i_2}, \Psi_{FS}^{i_3})$	$\mathcal{A}$ snoops on the communicated messages between SN, MV and FS with this query
$CorruptSD(\Psi_{SN_i}^{i})$	$\mathcal{A}$ can acquire all the accumulated secret credentials of any smart device SN that is compromised with this query
$Reveal(\Psi^i)$	$\mathcal{A}$ uses this query to acquire the common session key between Ψ^c and the participant it is communicating with
$Test(\Psi^i)$	$\mathcal{A}$ substantiates whether the established session keys between SN and MV, and between MV and FS are random or not using this query

authentication phase and the session key $SK_{MF} = (SK_{FM})$ between a mobile vehicle MV and the associated fog server FS in the $MVFS$ authentication phase of the proposed AgroMobiBlock in a particular session". Then, $Adv_{\mathcal{A}}^{AgroMobiBlock}(t_p) = |2Pr[b' = b] - 1|$, where b and b' correspond to the "correct" and "guessed" bits, respectively.

2) **Random oracle model**: In this section, we now apply the above discussed random oracle model to prove that the scheme provides the session key security according to the security model defined above.

Theorem 7.1. *Assume that a probabilistic polynomial time (PPT) adversary $\mathcal{A}$ taking t_p execution time attempts to obtain the session key SK_{SMV} $(= SK_{MVS})$ established between smart device SN and mobile vehicle MV during the $SNMV$ authentication phase, and the session key SK_{MF} $(= SK_{FM})$ established amid mobile vehicle MV and fog server FS for a given session during the $MVFS$ authentication phase of the proposed $AgroMobiBlock$. If q_h, $|Hash|$ and $Adv_{\mathcal{A}}^{ECDDHP}(t_p)$ represent the "number of $Hash$ queries", the "range space of a one-way collision-resistant hash function $H(\cdot)$" and the "advantage in breaking the Elliptic Curve Decisional Diffie-Hellman Problem (ECDDHP)", respectively, then $Adv_{\mathcal{A}}^{AgroMobiBlock}(t_p) \leq \frac{q_h^2}{|Hash|} + 2Adv_{\mathcal{A}}^{ECDDHP}(t_p)$.*

Proof. In $AgroMobiBlock$, the adversary $\mathcal{A}$ executes three games $Game_i^{\mathcal{A}}$, $(i = 0, 1, 2)$ such that in each game if $\mathcal{A}$ can guess a random bit b in $Game_i^{\mathcal{A}}$ accurately, the event $Success_{Game_i}^{\mathcal{A}}$ is said to be occurred. The probability that $\mathcal{A}$ is successful in guessing

the correct bit and wins $Game_i^{\mathcal{A}}$ is the advantage of $\mathcal{A}$ given as $Adv_{\mathcal{A},Game_i}^{AgroMobiBlock} = Pr[Success_{Game_i}^{\mathcal{A}}]$. The following games are designed to be played by the adversary $\mathcal{A}$ against the proposed $AgroMobiBlock$:

- $Game_0^{\mathcal{A}}$: In this game, $\mathcal{A}$ picks a random bit b before the game $Game_0^{\mathcal{A}}$ starts in order to launch an actual attack against $AgroMobiBlock$. By the "semantic security" definition, it follows that

$$Adv_{\mathcal{A}}^{AgroMobiBlock}(t_p) = |2Adv_{\mathcal{A},Game_0}^{AgroMobiBlock} - 1| \tag{7.1}$$

- $Game_1^{\mathcal{A}}$: In this game, $\mathcal{A}$ runs the $Execute$ query first and then the $Test$ query to launch an eavesdropping attack. The $Test$ query allows $\mathcal{A}$ to verify if the key values resulting from the $Reveal$ query are the actual session keys or some random keys. Using the $Execute$ query, $\mathcal{A}$ reads the messages Msg_{SM_1}, Msg_{SM_2} and Msg_{SM_3} during SNMV authentication phase and Msg_{MF_1}, Msg_{MF_2} and Msg_{MF_3} during MVFS authentication phase. The session key between smart device SN and mobile vehicle MV is $SK_{SMV} = H(i_S||\ TID_{SND}||\ RID_{SND}||\ pr_S||\ TS_S)$ $\cdot J_M = H(j_M||\ TID_M||\ RID_M||\ pr_M||\ TS_M)\ \cdot I_S = SK_{MVS}$. This session key consists of both the temporal (short-term) secrets (i_S, j_M) and long-term secrets (RID_S, RID_M, pr_S, pr_M). Moreover, the session key $SK_{FM} = H(H(K_M||\ TS_{mc})$ $||\ U_M\ ||\ V_F\ ||\ H(K_F||TS_{fc})\ ||\ DK_{FM}) = SK_{MF}$ where $DK_{FM} = H(v_F||\ pr_F||$ $TID_F||\ TS_{mf2})\ \cdot\ U_M = H(u_M||\ pr_M||\ TID_M||\ TS_{mf1})\ \cdot\ V_F$ also depends on temporal (short-term) secrets (u_M, v_f) and long-term secrets (K_M, K_F, pr_M, pr_F). The secret parameters are further secured using a "collision-resistant one-way hash function $h(\cdot)$". The session keys SK_{SMV} $(= SK_{MVS})$ and SK_{MF} $(= SK_{FM})$ are not revealed even if $\mathcal{A}$ directly captures the communicated message and it does not result in any increase of the success probability of $Game_1^{\mathcal{A}}$. Therefore, $Game_0^{\mathcal{A}}$ and $Game_1^{\mathcal{A}}$ cannot be distinguished under an eavesdropping attack. Thus, it follows that

$$Adv_{\mathcal{A},Game_1}^{AgroMobiBlock} = Adv_{\mathcal{A},Game_0}^{AgroMobiBlock}. \tag{7.2}$$

- $Game_2^{\mathcal{A}}$: Under this game, $\mathcal{A}$ simulates the $Hash$ queries and the computational ECDDHP problem to launch an active attack. In the SNMV authentication phase, the session key is derived as $SK_{SMV} = H(i_S||\ TID_{SND}||\ RID_{SND}||\ pr_S||\ TS_S)$ $\cdot J_M = SK_{MVS}$. $\mathcal{A}$ can obtain I_S and J_M directly from Msg_{SM_1} and Msg_{SM_2} in communication. $\mathcal{A}$ can derive the session key from publicly available I_S and J_M only if it can solve the computational ECDDHP that gives the private $H(i_S||$

$TID_{SND}||\ RID_{SND}||\ pr_S||\ TS_S)$ and $H(j_M||\ TID_M||\ RID_M||\ pr_M||\ TS_M)$ which are based on the secrets (i_S, j_M) that are unknown to $\mathcal{A}$. Then a simulation of the $Hash$ queries is needed in order to compute i_S and j_M that are protected by the "collision-resistant one-way hash function $H(\cdot)$". In the MVFS authentication phase, the secret parameters K_M, K_F are needed to compute $H(K_M||\ TS_{mc})$ and $H(K_F||TS_{fc})$ contributed by the TRA. $\mathcal{A}$ needs to simulate the $Hash$ queries to obtain $H(RID_M||\ TID_M||\ Pub_F||\ K_{MV_i,FS_j}||\ TS_{mf1})$ and $H(RID_F||\ TID_F||\ Pub_M||\ K_{MV_i,FS_j}||\ TS_{mf2})$ which require the secrets u_M, v_F and K_{MV_i,FS_j} unknown to $\mathcal{A}$. Then, the computational ECDDHP needs to be solved to extract $H(u_M||\ pr_M||\ TID_M||\ TS_{mf1})$, $H(v_F||\ pr_F||\ TID_F||\ TS_{mf2})$ from known U_M, V_F and G. Thus, $\mathcal{A}$ has to solve both $Hash$ queries and ECDDHP in order to derive the session keys in SNMV and MVFS phases. Exclusion of simulation of the $Hash$ queries and the computational ECDDHP from $Game_2^{\mathcal{A}}$ leads to indistinguishability of the games $Game_1^{\mathcal{A}}$ and $Game_2^{\mathcal{A}}$. Using the birthday paradox for finding the hash collision along with the advantage of solving ECDDHP gives:

$$\begin{aligned} &|Adv_{\mathcal{A},Game_1}^{AgroMobiBlock} - Adv_{\mathcal{A},Game_2}^{AgroMobiBlock}| \\ &\leq \frac{q_h^2}{2|Hash|} + \ Adv_{\mathcal{A}}^{ECDDHP}(t_p) \end{aligned} \tag{7.3}$$

All the queries except guessing a bit are executed by $\mathcal{A}$ to win the game $Game_2^{\mathcal{A}}$, leading to $Adv_{\mathcal{A},Game_2}^{AgroMobiBlock} = \frac{1}{2}$.

Eqs. (7.1), (7.2) and (7.3), along with the triangular inequality produce the following derivation:

$$\begin{aligned} \frac{1}{2}.Adv_{\mathcal{A}}^{AgroMobiBlock}(t_p) &= |Adv_{\mathcal{A},Game_0}^{AgroMobiBlock} - \frac{1}{2}| \\ &= |Adv_{\mathcal{A},Game_1}^{AgroMobiBlock} - Adv_{\mathcal{A},Game_2}^{AgroMobiBlock}| \\ &\leq \frac{q_h^2}{2|Hash|} + Adv_{\mathcal{A}}^{ECDDHP}(t_p). \end{aligned}$$

The following final result is achieved by multiplying both sides by "a factor of 2": $Adv_{\mathcal{A}}^{AgroMobiBlock}(t_p) \leq \frac{q_h^2}{|Hash|} + 2Adv_{\mathcal{A}}^{ECDDHP}(t_p)$. □

Remark 7.2 (Unforgeability of signatures)**.** *The signatures Sig_S, Sig_M generated in $SNMV$ phase and Sig_M, Sig_F generated in $MVFS$ phase are partitioned signatures as defined in [85] generated using the ElGamal signatures and hashing concept using*

the Secure Hash Algorithm (SHA-256). They generate the private random secrets i_S, j_M, u_M, $v_F \in Z_p^*$, apply one-way collision resistant hash function and elliptic curve multiplication to obtain I_S, J_M, U_M and V_F, which form the first part of the signatures. Then, the second part of the signature is generated by adding the private random secrets to the hash of the public parameters multiplied by their respective private keys. The unforgeability of ElGamal signature is proven in [256] and the collision resistance of SHA-256 is also proven in [89, 107, 148]. Hence, the signatures generated in both the $SNMV$ phase and $MVFS$ phase are unforgeable.

7.5.2 Informal security analysis

In this section, we show that the proposed *AgroMobiBlock* is also resilient against various attacks.

1) **Replay attack**: The messages Msg_{SM_1}, Msg_{SM_2} and Msg_{SM_3} in SNMV phase and Msg_{MF_1}, Msg_{MF_2} and Msg_{MF_3} in MVFS phase are transmitted through open channels, whereas the timestamps TS_S, TS_M, TS_{SM} are used in SNMV phase and the timestamps TS_{mf1}, TS_{mf2}, TS_{mf3} are used in MVFS phase. In addition, the random secrets i_S, j_M, u_M and v_F, respectively, are used in the parameters I_S, J_M, U_M and V_F sent through the above messages. The receiver verifies each timestamp before processing the message contents. During signature verification, the public parameters consisting of the random secrets are also verified. Due to this, the receiver can immediately identify if an adversary $\mathcal{A}$ captures a message and replays it. Thus, the proposed *AgroMobiBlock* is resilient against replay attack.

2) **Man-in-the-Middle (MiTM) attack**: In the SNMV phase, any changes to the parameters I_S, TID_{SND}, and RID_{SND} in Msg_{SM_1}, J_M, TID_M, RID_M, and TID^*_{SND} in Msg_{SM_2}, and TID^*_M and SKV_{SMV} in Msg_{SM_3} are easily identified in the signature verification of Sig_S and Sig_M or session key verification of SKV_{SMV} as SKV_{MVS} at the receiver. Similarly, in the MVFS phase, any changes to the parameters U_M, TID_M, RID_M, and K^h_M in Msg_{MF_1}, V_F, TID_F, RID_F, and K^h_F in Msg_{MF_1}, and SKV_{mf} in Msg_{MF_1} can be easily identified in the signature verification of Sig_M and Sig_F or session key verification of SKV_{mf} as SKV_{fm} at the receiver. Thus, MiTM attack is resilient in the proposed *AgroMobiBlock*.

3) **Impersonation attacks**: Consider the assumption that an adversary has intercepted the messages Msg_{SM_1}, Msg_{SM_2} and Msg_{SM_3} in the SNMV phase and Msg_{MF_1}, Msg_{MF_2} and Msg_{MF_3} in the MVFS phase. The following are the possible impersonation attacks:

   - *IoT smart device impersonation attack*: Consider that the adversary $\mathcal{A}$ tries to impersonate the smart device SN in the SNMV phase. It has to fabricate a fake message $Msg_{SM_1}^{adv} : \langle I_S^{adv}, TID_{SND}^{adv}, RID_{SND}^{adv}, TS_S^{adv}, Sig_S^{adv} \rangle$ for which it has to generate the random secret i_S^{adv} to compute I_S^{adv} and the timestamp TS_S^{adv}. However, this requires $\mathcal{A}$ to know the secrets i_S, ID_{SND}, and s. Since these secrets are never shared in any message, the IoT smart device impersonation attack is not possible in the proposed scheme.
   - *Mobile vehicle impersonation attack*: Consider that $\mathcal{A}$ tries to impersonate the mobile vehicle MV in the SNMV and MVFS phases. The adversary $\mathcal{A}$ requires the knowledge of j_M, ID_M, and m to fabricate the message $Msg_{SM_2}^{adv}$: $\langle J_M^{adv}, Sig_M^{adv}, TID_M^{adv}, RID_M^{adv}, TID_{SND}^{*adv}, TS_M^{adv} \rangle$ in SNMV phase, and m, RID_M from TRA, u_M, ID_M from MV, along with K_M^* stored on the blockchain in order to fabricate the message $Msg_{MF_1}^{adv}$: $\langle U_M^{adv}, TID_M^{adv}, RID_M^{*adv}, Sig_M^{adv}, K_M^{h-adv}, TS_{mf1}^{adv} \rangle$ leading to conclude that the proposed scheme is resilient to mobile vehicle impersonation attack.
   - *Fog server impersonation attack*: Consider that $\mathcal{A}$ tries to impersonate the fog server FS in the MVFS phase. It has to fabricate a fake message $Msg_{MF_2}^{adv} : \langle V_F^{adv}, Sig_F^{adv}, TID_M^{*adv}, TID_F^{adv}, RID_F^{*adv}, K_F^{h-adv}, TS_{mf2}^{adv} \rangle$ and requires the knowledge of the private secrets f, RID_F from TRA, ID_F, v_F from FS, along with K_F^* stored on the blockchain. Hence, *AgroMobiBlock* is resilient against fog server impersonation attacks.

4) **Privileged-insider attack**: All the required secret credentials during the registration of the IoT smart device's ID_S, s, the mobile vehicle's ID_M, m, K_M and the fog server ID_F, f, K_F are either pre-loaded into the memory of the corresponding entity by the TRA or passed through a secure channel or not used directly. Thus, *AgroMobiBlock* is strongly resilient against privileged insider attack.

5) **Physical IoT smart device and mobile vehicle capture attacks**: An adversary $\mathcal{A}$ who captures an IoT smart device SN can extract the information $\{(RID_{SND}, TID_{SND}), H(\cdot), E_q(\kappa, \mu), G, (pr_S, Pub_S)\}$ using power analysis attacks [200] and

timing attacks [199]. A compromised smart device cannot affect the communication among the non-compromised smart nodes as none of the devices share any secret credentials or established session keys. Such a compromised node is replaced with a new node using the dynamic node addition phase as described in Section 7.4.6. Thus, *AgroMobiBlock* is resilient against the physical IoT smart device capture attack. The adversary $\mathcal{A}$ cannot obtain any information from the captured mobile vehicle's memory as the blockchain stores its secret credentials securely. Hence, a physical capture attack on a mobile vehicle fails.

6) **Ephemeral secret leakage (ESL) attack**: The session key in the SNMV phase, $SK_{SMV} = SK_{MVS}$, depends both on the temporal secrets i_S and j_M, and long term secrets TID_{SND}, RID_{SND}, TID_M, RID_M, pr_S and pr_M. Similarly, the session key in the MVFS phase, $SK_{FM} = SK_{MF}$, requires the dependency on the short-term secrets u_M and v_F, and the long term secrets K_M, K_F, TID_M, RID_M, TID_F, RID_F, pr_M and pr_F. We now consider the following two scenarios:

- If only the short-term secrets (i_S, j_M) in the SNMV phase and (u_M, v_F) in the MVFS phase are compromised, the session key $SK_{SMV} = SK_{MVS}$ is still uncompromised as $\mathcal{A}$ does not have access to the long term secrets (TID_{SND}, RID_{SND}, TID_M, RID_M, pr_S and pr_M) in SNMV phase and (K_M, K_F, TID_M, RID_M, TID_F, RID_F, pr_M and pr_F) in MVFS phase. Moreover, the secrets K_M, and K_F are only used in the hash computation and are never shared anywhere during the registration or authentication phase.
- If only the long term secrets (TID_{SND}, RID_{SND}, TID_M, RID_M, pr_S and pr_M) in SNMV phase and (K_M, K_F, TID_M, RID_M, TID_F, RID_F, pr_M and pr_F) in MVFS phase are compromised, the session key $SK_{SMV} = SK_{MVS}$ is still non-compromised as $\mathcal{A}$ does not have access to the short-term secrets are (i_S, j_M) in SNMV phase and (u_M, v_F) in MVFS phase.

For the adversary $\mathcal{A}$ to succeed in the ESL attack, the short-term and long-term secrets need to be compromised together. The proposed *AgroMobiBlock* is then strongly resistant to ESL attacks under the CK adversary threat model.

7) **Denial of service (DoS) attack**: An attacker can launch a DoS attack to impede access to a server. To launch a DoS attack, the adversary accesses the server and consumes its resources to render them unusable for any other purpose. DoS attacks can be averted by preventing the adversary from gaining access to the server. After

receiving Msg_{MF_1}, if verification of Sig_M fails at FS, the entity MV attempting to access FS will be deemed as not authentic and access to the fog server will be thwarted. Thus, *AgroMobiBlock* is resistant to DoS attack.

8) **Advanced persistent threat**: An adversary slowly intrudes into the network to ex-filtrate specific targeted sensitive data using techniques, such as malware, social engineering, and zero-day vulnerabilities [44, 61, 343]. The proposed system uses the blockchain to actively store the authentication credentials and sensor data in separate blocks. The inherent capabilities of a blockchain including tamper-proof storage, secure the stored data. In addition, the SensorData block stores the actual sensor data in an encrypted format. With no access to the private key of the fog server, the intruding group cannot access the sensor data as well.

9) **Anonymity and untraceability**: The blockchain inherently possesses the characteristics of anonymity and untraceability on the data stored in the block. However, the authentication process must also satisfy the anonymity and untraceability properties separately. In the $SNMV$ phase, the messages Msg_{SM_1}, Msg_{SM_2}, and Msg_{SM_3} use only the temporary identities TID_{SND}, TID_M and hidden TID_M^*, and the pseudo-identities RID_{SND}, and RID_M instead of the original identities ID_{SND} and ID_M. Similarly, the $MVFS$ phase only uses the temporary identities TID_M and TID_F with the hidden pseudo-identities RID_M^* and RID_F^* instead of the original identities ID_M and ID_F. Thus, none of the messages can be traced back to the original identities of the sender. This preserves the anonymity of all the entities. The untraceability feature makes the public details in the messages unlinkable. Due to this, an attacker cannot obtain any information from the traffic in the public channel. Since none of the messages have any common contents, any parameters of a message cannot be inferred from other messages.

7.6 Formal security verification using AVISPA: simulation study

This section focuses on the simulation of the proposed scheme (*AgroMobiBlock*) using the widely regarded tool "Automated Validation of Internet Security Protocols and Applications (AVISPA)" that validates the protocol as a safe protocol or unsafe protocol

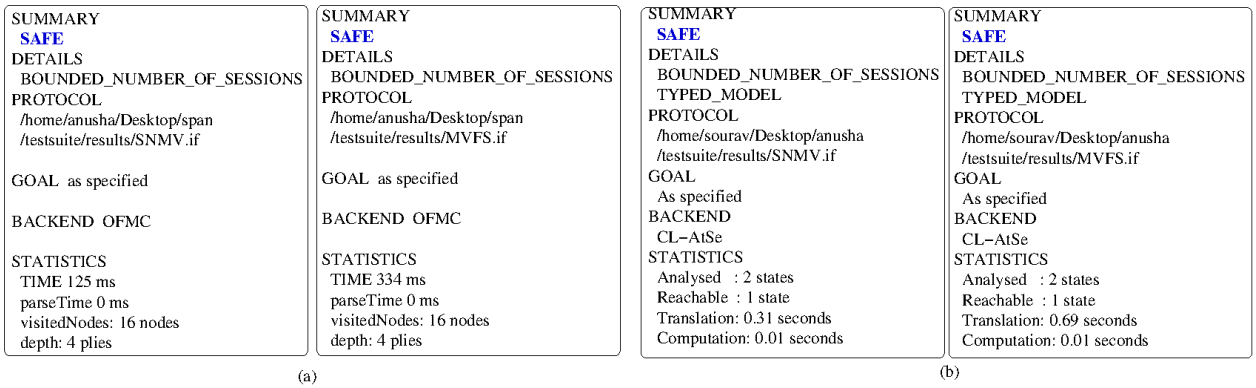

Figure 7.10: Simulation results of *AgroMobiBlock* for Case 1 and Case 2: (a) OFMC backend (b) CL-AtSe backend

against replay attacks and man-in-the-middle attacks. We have considered two cases: Case 1- authentication between SN and MV and Case 2- authentication between MV and FS. It is then simulated through the "SPAN, the Security Protocol ANimator for AVISPA" [22] on "On The Fly Model Checker (OFMC)" and "Constraint Logic based Attack Searcher(CL-AtSe)" backends. Details on the AVISPA tool can be found in Appendix A.

The proposed scheme is coded in "high level protocol specification language (HLPSL)". It consists of four roles *rauthority*, *sensor*, *mobiledev*, and *fog* corresponding to the trusted registration authority (TRA), sensor node (SN), mobile vehicle (MV), and fog server (FS) entities, respectively. The roles for the session and environment are created. Each role consists of four agents for each of the entities, a hash function, two Dolev-Yao channels corresponding to send and receive, and a symmetric key to encrypt the communication privately. The agent to play every role is assigned. Whenever a role receives a message, it changes its state. There is no change of state when a message is sent. For every message, the sender creates a witness over the timestamp and private secret while the receiver creates a request. The environment role makes a set of parameters available to the intruder. It composes the session for the replay attack by invoking multiple sessions with all honest entities and for MiTM attack by invoking multiple sessions with the intruder acting as one of the honest entities for each invocation. Protocols are defined on random secret parameters with a secrecy goal. For the timestamps and secrets on which request-witness is applied, the authentication goal is defined.

The results presented in Figure 7.10(a) under the OFMC backend and in Figure 7.10(b)

under the CL-AtSe backend clearly illustrate that the proposed scheme is safe against passive/active attacks, like replay and man-in-the-middle attacks.

Table 7.3: Comparison of communication costs

Protocol	No. of messages	Total cost (in bits)
Ali *et al.* [52]	5	5504
Chen *et al.* [102]	4	4960
Chae and Cho [96]	4	12896
Rangwani *et al.* [265]	5	4128
Wu and Tsai [340]	10	$1344 + 256n$
SCBAS-SF [Chapter 6]	6	4576
AKMS-AgriIoT [Chapter 5]	4	3456
Tian *et al.* [311]	2	$384s + 11712$
Shuai *et al.* [286]	4	7616
Panda *et al.* [253]	5	$2N * 256 + 4480$
Eddine *et al.* [130]	4	2273
Fan *et al.* [133]	5	4480
Tomar & Tripathi [312]	3	4112
Itoo *et al.* [174]	4	1408
Jia *et al.* [179]	6	3616
AgroMobiBlock (Phase-1)	3	2848
AgroMobiBlock (Phase-2)	5	4608

Note: s: "number of pseudonyms of a drone in Tian *et al.*'s scheme" [311]; N: "a positive integer that specifies the number of public/private key pairs available to an IoT device in Panda *et al.*'s scheme" [253]; n: "number of agricultural equipment (sensor devices) in Wu and Tsai's scheme" [340].

7.7 Comparative study

This section compares the performance of the proposed *AgroMobiBlock* on the costs of communication and computation for the SNMV authentication and key agreement phase (Phase 1) and the MVFS authentication and key agreement phase (Phase 2). These costs

Table 7.4: Comparison of computation costs

Protocol	Smart device/Drone end	GSS/Server end
Ali *et al.* [52]	$11T_h + T_{fe} + 3T_{senc}/T_{sdec}$ $\approx$ 5.738 ms	$8T_h + 5T_{senc}/T_{sdec}$ $\approx$ 0.445 ms
Chen *et al.* [102]	20 $T_h \approx$ 6.18 ms	17 $T_h \approx$ 0.935 ms
Chae and Cho [96]	8 T_{ecm} + 8 T_h + $2Teca \approx$ 20.808 ms	–
Rangwani *et al.* [265]	8 T_h + 5 $T_{ecm} \approx$ 13.912 ms	7 T_h + $T_{ecm} \approx$ 1.059 ms
Wu and Tsai [340]	$2T_{bp}$ + $2T_{exp}$ + $2T_{senc/sdec}$ + $1T_h \approx$ 64.965 ms	$2T_{bp}$ + $2T_{exp}$ + $2T_{senc/sdec}$ + $1T_h \approx$ 9.407 ms
SCBAS-SF [Chapter 6]	$17T_h + 12T_{ecm} \approx$ 32.645 ms	$9T_h + 4T_{ecm} \approx$ 3.191 ms
AKMS-AgriIoT [Chapter 5]	$18T_h + 14T_{ecm} + 2T_{eca} + T_{poly}$ $\approx$ 39.758 ms	$7T_h + 6T_{ecm} + 2T_{eca} + T_{poly} \approx$ 4.733 ms
Tian *et al.* [311]	$8T_{exp} + 9T_h \approx$ 4.605 ms	–
Shuai *et al.* [286]	$13T_h + 3T_{exp} \approx$ 4.701 ms	$7T_h + T_{exp} \approx$ 0.457 ms
Panda *et al.* [253]	$(N+k+l+3)T_h + N*k*NT_h$ $\approx$ 110 ms	$4T_{senc} + 2T_{sdec}$ $\approx$ 10.915 ms
Eddine *et al.* [130]	$3T_h$ + $3T_{ecm}$ + T_{ECCEnc} + $T_{ECCDec} + T_{CKDf} \approx$ 19.279 ms	$2T_H$+ $3T_{ecm}$+ $2T_{ECCEnc}$+ T_{ECCDec}+ $T_{CKDf} \approx$ 6.858 ms
Fan *et al.* [133]	$4T_{bp} + 5T_h + 6T_{ecm} +$ $2T_{eca} + 3T_{senc} + T_{sdec}$ $\approx$ 143.709 ms	$4T_{bp} + 5T_h + 6T_{ecm} +$ $2T_{eca} + 2T_{senc} + T_{sdec}$ $\approx$ 22.738 ms
Tomar & Tripathi [312]	$4T_{ecm} + 7T_h \approx$ 11.315 ms	$13T_{ecm} + 15T_h + 4T_{eca} \approx$ 9.595 ms
Itoo *et al.* [174]	$5T_{ecm}$ + $3T_{eca}$ + $8T_h \approx$ 13.96 ms	$7T_{ecm}$ + $5T_{eca}$ + $11T_h \approx$ 5.333 ms
Jia *et al.* [179]	$2T_{ecm} + 13T_h \approx$ 8.593 ms	$4T_{ecm} + 13T_h \approx$ 3.411 ms
AgroMobiBlock (Phase 1)	$6T_h$+ $4T_{ecm}$+ T_{eca} $\approx$ 11.022 ms	$6T_H$+ $4T_{ecm}$+ T_{eca} $\approx$ 3.028 ms
AgroMobiBlock (Phase 2)	–	$24T_h$+$8T_{ecm}$+ $2T_{eca}$+$2T_{ECCDec}$ $\approx$ 8.068 ms

Note: k,l: "k^{th} and l^{th} devices communicate with each other such that $0 < k, l < N$ in Panda *et al.*'s scheme"; N: "a positive integer that specifies the number of public/private key pairs available to an IoT device in Panda *et al.*'s scheme". Consider $N = 20$, $k = 5$ and $l = 7$.

are studied against the schemes developed by Ali *et al.* [52], Chen *et al.* [102], Chae and Cho [96], Rangwani *et al.* [265], Wu and Tsai [340], SCBAS-SF [Chapter 6], AKMS-

AgriIoT [Chapter 5], Tian *et al.* [311], Shuai *et al.* [286], Panda *et al.* [253], Eddine *et al.* [130], Fan *et al.* [133], Tomar and Tripathi [312], Itoo *et al.* [174] and Jia *et al.* [179] along with a thorough analysis of their security and functionality features.

7.7.1 Communication costs comparison

This section focuses on comparing the proposed scheme with the existing schemes in terms of the amount of cost expended in the communication of messages. The proposed *AgroMobiBlock* scheme takes 2848 bits for communication in the SNMV phase and 4608 bits in the MVFS phase. Even with the exchanged transaction data in the first message of MVFS phase, the total communication cost only accounts for 4608 bits, which is reasonably comparable with respect to the other schemes. Table 7.3 compares the communication costs during exchange of messages in the proposed scheme (*AgroMobiBlock*) and other existing schemes. It can be observed that *AgroMobiBlock* takes comparable communication costs to achieve more security and functionality features.

7.7.2 Computation costs comparison

The proposed scheme is compared with the existing schemes in terms of the amount of cost expended in the computation of the operations involved in the scheme. We have used the broadly accepted "Multiprecision Integer and Rational Arithmetic Cryptographic Library (MIRACL)" [38] to evaluate the execution time of various cryptographic primitives. The experiments for each cryptographic primitive are provided in Appendix B.

The proposed *AgroMobiBlock* scheme takes $6T_h + 4T_{ecm} + T_{eca}$ computation cost at the IoT smart node and the same cost $6T_h + 4T_{ecm} + T_{eca}$ at the mobile vehicle in the SNMV phase. It takes $24T_h + 8T_{ecm} + 2T_{eca} + 2T_{ECCDec}$ at both the mobile vehicle and fog server in the MVFS phase. Encryption and decryption do not lead to performance bottlenecks as they are only performed by mobile vehicles and fog servers, which are resource abundant. Encryption is only done once for each one-time registration of MV and FS. Table 7.4 compares the computation costs during the exchange of messages in the proposed scheme (*AgroMobiBlock*) and other existing schemes. It can be observed that *AgroMobiBlock* takes a comparable computation cost to achieve more security functionality features.

Table 7.5: Comparison of blockchain-related features

Scheme	Blockchain purpose	Usage of blockchain during registration	authentication	storing sensor data	storing secrets
Wu and Tsai *et al.*	*data packets distributed over multiple blockchains *dark web hides physical location of blockchain servers	×	×	✓	×
Vangala *et al.*	*store sensor data after authentication	×	×	✓	×
Bera *et al.*	*store sensor data after authentication	×	×	✓	
Eddine *et al.*	*store authentication results	×	×	×	×
Fan *et al.*	*store sensor data after authentication	×	×	✓	×
Tomar & Tripathi	*dual blockchain architecture with cloud and fog servers *multiple channels *stores public keys to extract identities	×	×	✓	×
Itoo *et al.*	*store authentication messages	×	✓	×	×
Jia *et al.*	*blockchain acts as PKI to store public keys and certificates	×	✓	✓	×
AgroMobiBlock	*store credentials during registration *access credentials during authentication *store sensor data on blockchain	✓	✓	✓	✓

7.7.3 Comparison of blockchain features

This section gives a comparison of the level of involvement of blockchain in the existing blockchain-based schemes. It is observed from Table 7.5 that even though some schemes access the blockchain during authentication to store public keys or authentication messages, none of them used the blockchain to store secrets during registration and later access them during authentication, along with storing sensor data at the same time. Our proposed scheme makes maximum use of the single blockchain by using it in multiple phases and storing credentials along with sensor data.

We analyze the usage of blockchain in the blockchain-based authentication schemes.The blockchain in [340] stores the packets received from the agricultural equipment. In [130], the authentication results are stored into the blockchain at the end of the authentication process. The schemes in [133], Chapter 6 and Chapter 7 store the data from IoT nodes on the blockchain, whereas the scheme in [174] stores all the messages exchanged during the authentication over blockchain. Even though the blockchain in [179] is accessed

Table 7.6: Comparison of security and functionality features

Scheme	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}
Authentication in Smart Agriculture without Blockchain														
Ali *et al.* [52]	×	×	✓	×	×	×	×	✓	✓	✓	✓	×	✓	NA
Chen *et al.* [102]	✓	✓	✓	✓	×	×	×	✓	✓	✓	✓	✓	✓	NA
Chae and Cho [96]	×	×	×	×	NA	×	×	×	×	✓	✓	×	×	NA
Rangwani *et al.* [265]	×	×	×	✓	✓	×	✓	✓	✓	✓	✓	×	✓	NA
Authentication in Smart Agriculture with Blockchain														
Wu and Tsai [340]	✓	✓	×	×	NA	×	✓	✓	✓	✓	✓	✓	×	×
SCBAS-SF [Chapter 6]	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	×
AKMS-AgriIoT [Chapter 5]	✓	✓	×	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	×
Authentication in Diverse Applications without Blockchain														
Tian *et al.* [311]	×	×	✓	×	✓	×	✓	✓	✓	✓	✓	×	×	NA
Shuai *et al.* [286]	✓	✓	✓	×	×	×	×	✓	✓	✓	✓	✓	×	NA
Panda *et al.* [253]	×	×	×	✓	NA	×	✓	✓	✓	✓	✓	✓	✓	NA
Authentication in Diverse Applications with Blockchain														
Eddine *et al.* [130]	×	×	×	✓	×	×	×	✓	✓	✓	✓	×	✓	×
Fan *et al.* [133]	×	×	×	✓	×	×	✓	✓	✓	✓	✓	×	×	×
Tomar & Tripathi [312]	✓	✓	×	✓	×	×	×	✓	✓	✓	✓	×	×	×
Itoo *et al.* [174]	×	×	NA	✓	NA	×	×	✓	✓	✓	✓	×	×	×
Jia *et al.* [179]	✓	✓	×	✓	×	×	×	✓	✓	✓	✓	×	×	×
DID Authentication														
Andola *et al.* [57]	✓	✓	NA	✓	×	×	×	×	×	✓	×	×	✓	NA
Liu *et al.* [220]	×	×	NA	×	×	×	×	✓	×	✓	✓	×	×	NA
Mishra *et al.* [240]	×	×	NA	×	×	×	×	✓	×	✓	✓	×	×	NA
Gupta *et al.* [154]	×	×	×	×	×	×	×	✓	×	×	×	×	×	NA
AgroMobiBlock	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Note: F_1 : "anonymity", F_2 : "untraceability", F_3 : "dynamic node addition", F_4 : 'device/user impersonation attacks", F_5 : "stolen mobile device attacks", F_6 : "ESL attacks", F_7 : "privileged insider attacks", F_8 : "replay attacks", F_9 : "MiTM attacks", F_{10} : "mutual authentication", F_{11} : "unauthorized login detection", F_{12} : "DoS attacks", F_{13} : "offline guessing attacks", F_{14} : "active blockchain (THIS WORK)".
✓: "Supports feature/resists attacks"; ×: "No support for feature/vulnerable to attacks"; NA: "Not applicable"

during authentication (since it uses the blockchain as an alternative form of "Public Key Infrastructure (PKI)" instead of storing critical secrets for the authentication process), it does not conform to the concept of active blockchain. It stores only public keys of all

entities, including the end users on the blockchain, via a smart contract. In addition, innumerable end users could make their blockchain large and unmanageable. Thus, these schemes use passive blockchain, which does not play any role during the authentication process. The current research aims to explore blockchain's usage in storing and retrieving crucial secrets while the authentication is in progress.

7.7.4 Security and functionality features comparison

Table 7.6 provides a comparative analysis on various "security and functionality features" among the proposed scheme (*AgroMobiBlock*) and other existing schemes in [52, 96, 102, 130, 133, 253, 265, 286, 311, 340]. The analysis reveals the need for the proposed *AgroMobiBlock* as it achieves more security and functionality features compared to the existing authentication schemes. Comparing the proposed scheme with the existing ones, it is observed that *AgroMobiBlock* has the ability to resist ESL attack while simultaneously achieving anonymity and untraceability properties for the IoT smart devices, mobile vehicles, and fog servers. In addition, the active blockchain presented in this work is not supported by any other relevant schemes without the need for smart contracts.

7.7.5 Discussion on performance analysis

AgroMobiBlock has lower communication cost compared to the schemes in [52, 96, 102, 253, 286, 311, 340], and significantly lower computation cost compared to [96, 130, 133, 253, 265, 340]. Wu and Tsai's scheme [340] is a blockchain-based scheme, but it incurs a very high cost for computation and communication, lacks dynamic node addition, and is vulnerable to ESL and offline guessing attacks. Rangwani *et al.*'s scheme [265] has low computation cost, but it still lacks anonymity, untraceability, dynamic node addition and blockchain support, and cannot resist ESL and DoS attacks. *AgroMobiBlock* achieves all the security and functionality features with low communication and computation costs, which is not demonstrated in any of the compared schemes.

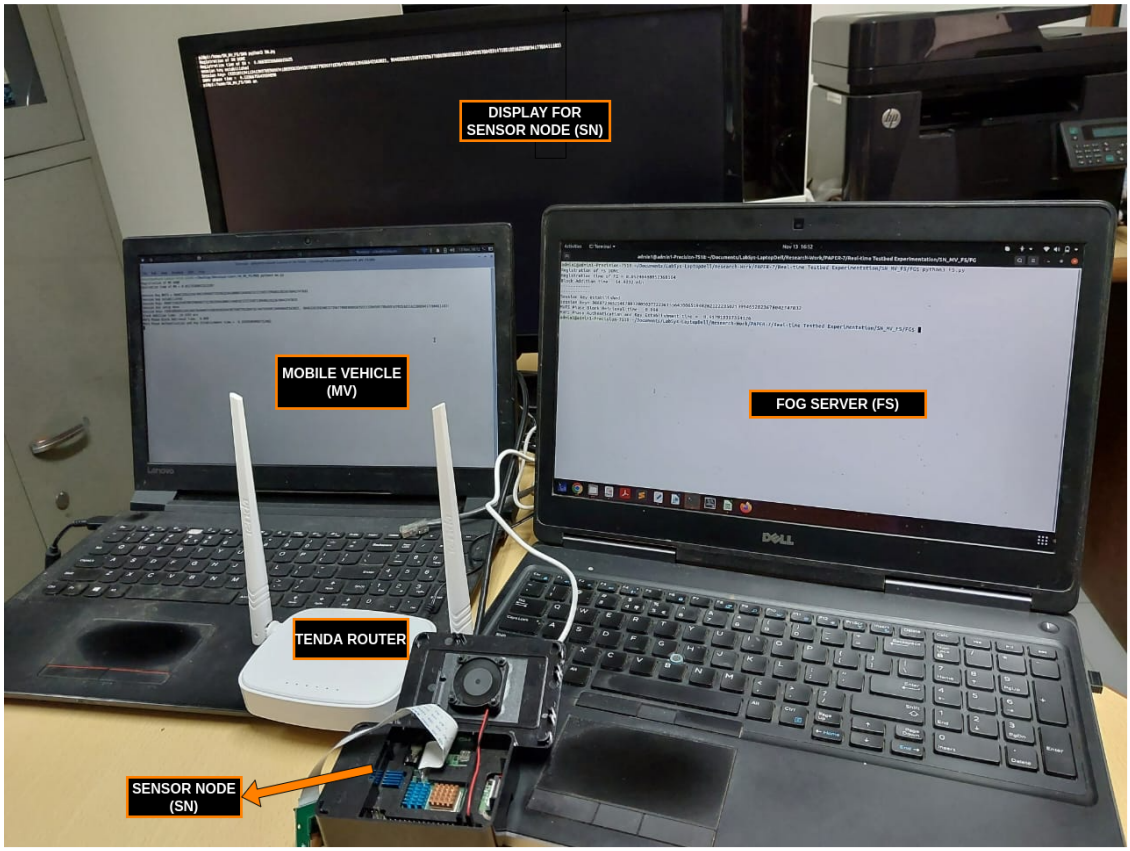

Figure 7.11: Experimental Testbed Setup

```
pi@pi:/home/SN$ python3 SN.py
***** REGISTRATION OF SENSOR NODE (SN) **********
Registration of SN is in progress.........
Registration of SN is completed.
Registration time taken for SN:  62.16764450073242  milliseconds

****** SNMV PHASE *****
SNMV phase begins....
Authentication of SN and MV is in progress....
Session key establishment between SN and MV is in progress....
Session key between SN and MV established successfully....
Session key between SN and MV established successfully.
Session key between SN and MV:
( 437382114278029168052816020431537152947526717919238507398209402241632544855 31 ,
90916468239859088909186972099441839784435361350137545761941166999896496568333 )
Time taken for SNMV phase: 401.0009765625  milliseconds
pi@pi:/home/SN$ _
```

Figure 7.12: Execution output from a sensor node (SN) side

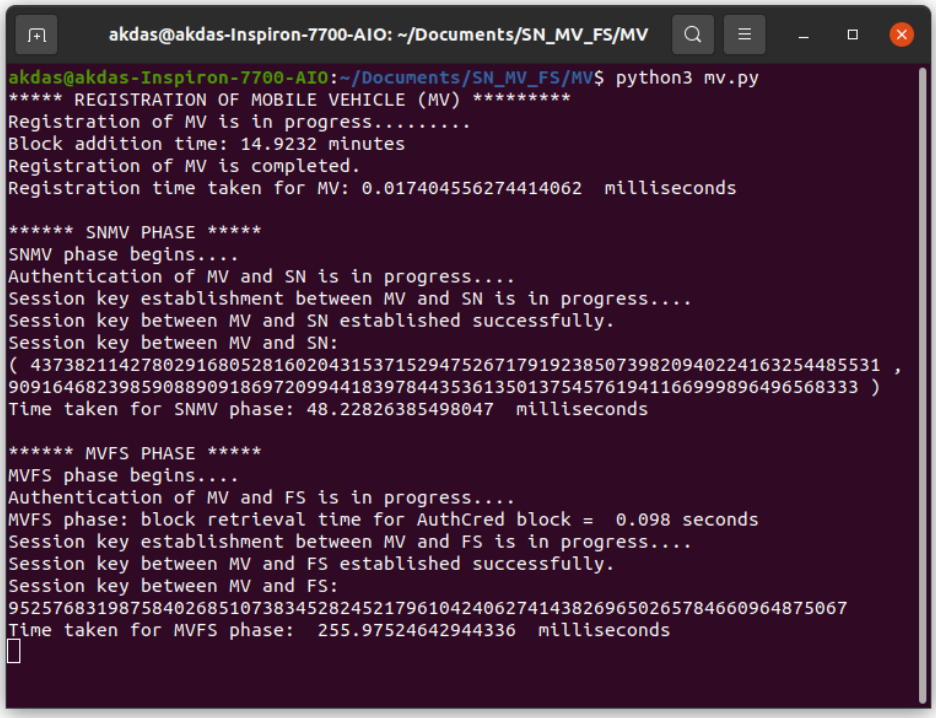

Figure 7.13: Execution output from a mobile vehicle (MV) side

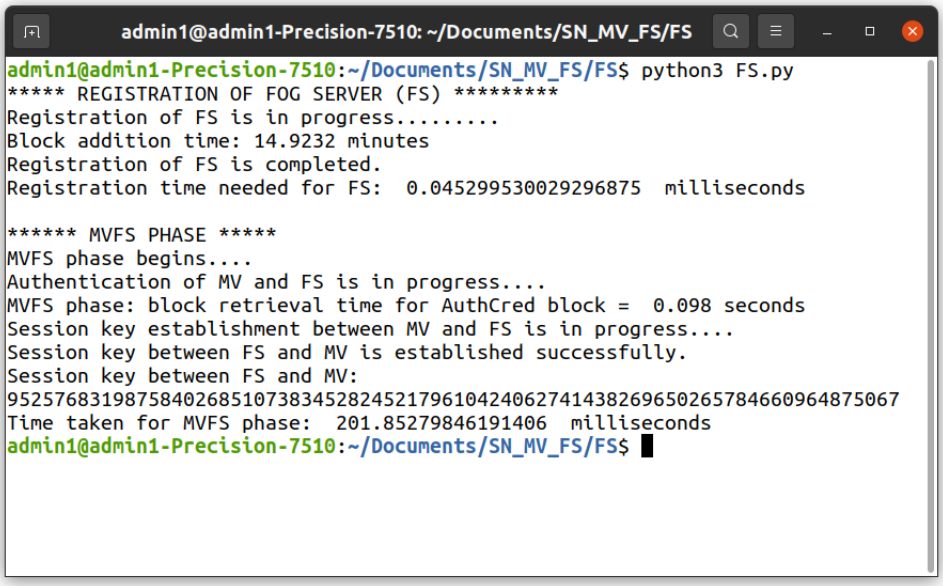

Figure 7.14: Execution output from a fog server(FS) side

7.8 Real-time practical implementation using testbed experimentation

This section presents the environment of the real-time testbed experimental setup that implements the proposed model with both the authentication and key agreement phase between a sensor node and mobile vehicle (SNMV phase), the authentication and key agreement phase between a mobile vehicle and a fog server (MVFS phase) as well as the registration phases as described in Section 7.4. A detailed description of the setup and the experimental results for this implementation are provided below.

7.8.1 Testbed setup

For the experimental testbed setup, we simulate the SNMV phase and MVFS phase. In the following, we have the setup for various entities as follows:

- A sensor node (SN) is implemented with 64-bit Ubuntu 20.04 LTS operating system installed over the Raspberry PI 4 hardware. A python script in the sensor node acts as a remote service on PI and communicates using the socket programming.
- A mobile vehicle (MV) is implemented using a system with the configuration: 16GiB RAM, eight 11^{th} generation Intel Core i7-1165G7 processors with 2.80GHz clock speed each and 64-bit Ubuntu 20.04 LTS operating system. A python script runs the mobile vehicle as a remote service, and the socket programming allows communication between a sensor node and a fog server.
- The fog server (FS) is then implemented using a laptop with 32GiB RAM, eight Intel Core i7-6820HQ processors with 2.70GHz clock speed each, and 64-bit Ubuntu 20.04 LTS operating system. A python script runs the fog server as a remote service, and the socket programming allows communication with a mobile vehicle.

We used the Tenda router to connect the nodes to the same private network. The private Internet Protocol (IP) address is used to establish the communication in the socket programming. Figure 7.11 provides the experimental testbed setup used in our implementation.

7.8.2 Experimental results and discussions

The output results of the *AgroMobiBlock* scheme shown in Figs. 7.12, 7.13 and 7.14 illustrate the execution time required for registration, block retrieval, SNMV phase and MVFS phase, in milliseconds.

The registration of a mobile vehicle needs 0.0174 milliseconds to register with the TRA, and 14.9232 minutes time is needed to add a block into the chain. A sensor node takes 62.1676 milliseconds to register with the TRA. The time taken for the $SNMV$ phase at the sensor node is 401 milliseconds, whereas it is 48.2282 milliseconds at the mobile vehicle. A fog server takes 14.9232 minutes to add a block into the chain and 0.0452 milliseconds to register with the TRA. In the $MVFS$ phase, it takes 0.098 seconds to retrieve an AuthCred block for the mobile vehicle and another AuthCred block for the fog server from the chain. The time taken for the $MVFS$ phase at the mobile vehicle is 255.9752 milliseconds and at the fog server the time needed is 201.8527 milliseconds.

The summary of execution time for testbed experiments (in milliseconds) for various phases related to the proposed scheme is provided in Table 7.7.

Table 7.7: Summary of execution time for testbed experiments

Phase	Execution Time (in milliseconds)
Sensor Node (SN)	
Registration phase	62.1676
Authentication with MV ($SNMV$ phase)	401
Mobile Vehicle (MV)	
Registration phase	0.0174
Authentication with SN ($SNMV$ phase)	48.2282
Authentication with FS ($MVFS$ phase)	255.9752
Fog Server (FS)	
Registration phase	0.0452
Authentication with FS ($MVFS$ phase)	201.8527

7.9 Blockchain implementation

As discussed in Section 7.4, the simulations assume that the fog servers associated with an agricultural field form a P2P fog system that receive transactions from mobile vehicles. The P2P fog system creates either an AuthCred block or SensorData block to be added to the single hybrid blockchain. There are 11 fog servers assumed to be in a P2P fog system. One of the fog servers, elected as the leader/miner/proposer in a round-robin fashion, initiates the PBFT consensus algorithm [95] for block creation, verification and inclusion into blockchain as discussed in detail in Algorithm 4.

The blockchain simulations were performed on a server platform having the environment: "CentOS Stream 8, with 64 CPUs, each with 64-bit OS with Intel(R) Xeon(R) Gold 6226R CPU @ 2.90GHz, 376 GiB RAM" using Node.js language with VS CODE 2019 [191]. The purpose of this simulation is to study the effect of an increase in the number of fog nodes, transactions, and blocks over the computational time.

In this simulation study, we have considered the synthetic data only. The IoT smart devices send the sensing data securely to their respective mobile vehicles and then the mobile vehicles also send the data securely to their nearby fog servers. The servers aggregate the genuine data as transactions, which are used to form blocks. Therefore, we do not consider any real-time agricultural-related data in this work.

The following scenarios are considered.

- **Scenario 1:** Here, we measure the time to add a single block and monitor its variation with the network size, that is, the number of nodes in the network. We increase the network size from 30 to 80 server nodes. The graph in Figure 7.15(a) depicts the relation between single block addition time, that is, consensus time, with the size of the network. The graph clearly shows that as the network size increases the single block addition time increases significantly. A PBFT-based voting mechanism requires more communication cost of $O(C^{N_f})$ where C is the number of messages exchanged in the pre-prepare, prepare and commit phases executed by the N_f servers participating in the consensus [95]. More number of nodes leads to increased time to reach consensus. This cost is in addition to the communication cost computed for Algorithm 4 in Section 7.4.5.
- **Scenario 2:** Here, we measure the time to add a single block and monitor its variation with the number of transactions in a block. The network size is fixed to 80

Figure 7.15: Blockchain simulation results: (a) Scenario 1 (b) Scenario 2 (c) Scenario 3 (d) Scenario 4

server nodes and the blockchain length is fixed at 100 blocks. The transactions count is varied from 100 to 400 transactions per block. In the graph in Figure 7.15(b), it is clearly observed that an increase in the number of transactions leads to an increase in the consensus computational time. This is because as the number of transactions in a block increase, each node in the P2P network has to verify the signature, block hash and Merkle tree root for the increased number of transactions adding to the overall time taken by the consensus.

- **Scenario 3:** This scenario measures the throughput of the blockchain as the number of transactions processed per second. This is obtained as the ratio of the number of transactions per block to the average block time taken in seconds. Figure 7.15 (c) depicts the throughput of the blockchain with the increase in the number of transactions per block from 100 to 450. It is observed that the throughput increases

when the number of transactions per block is increased while the number of P2P nodes and the number of blocks mined are fixed at 80 and 100, respectively. As the number of transactions per block increases, the time taken to process each block also increases, leading to higher number of transactions being processed per second over the entire network.

- **Scenario 4:** Here, the service time of a node in the blockchain is the time that a node is busy in processing the transactions of the block it received. A node places its transactions in a memory pool. If a transaction of a block being mined is unavailable in the memory pool, it requests the network to obtain it. To obtain the service time of a mining node, the ratio of the number of mining nodes to the time to mine each block is divided by the number of transactions per block. Figure 7.15 (d) shows that the service time of the node decreases as the number of transactions per block decreases.

In summary, a single block addition time, that is, consensus time is dependent on the size of the P2P network, but not on block size or blockchain length. Power consumption of Algorithm 4 is directly proportional to the algorithm execution time. Therefore, the blockchain simulation results show that the execution time also directly reflect the power consumption of the algorithm.

7.10 Summary

We designed a robust authenticated key agreement scheme using a hybrid blockchain with the help of mobile vehicles for a precision agricultural IoT network. The active blockchain is accessed to extract credentials for key agreement between a mobile vehicle and a fog server. Fog servers securely aggregate sensor data that form blocks of transactions. Blocks are mined with the help of a voting-based PBFT consensus mechanism. Detailed security analysis and comparative study reveal that the proposed scheme resists various attacks, offers more functionality attributes, and has comparable communication and computational costs to other competing schemes. Finally, the real-time testbed experiments are performed to exhibit the proposed scheme's practical usage.

Chapter 8

Conclusion and Future Research Directions

This chapter summarizes the major contributions of the thesis. In addition, we propose some of the unsolved challenges and open research directions in the area of smart agriculture.

8.1 Contributions

This thesis proposes four new protocols for achieving user authentication and mutual authentication using blockchain technology in a smart agricultural environment. These protocols are as listed below:

- Signature-based user authentication in a smart agricultural environment
- Private blockchain-based mutual authentication scheme using drone technology
- Hybrid blockchain-based mutual authentication scheme using smart contracts
- Hybrid blockchain-based mutual authentication scheme using mobile vehicles.

The first contribution of this thesis is provided in **Chapter** 4. It proposes a user authentication scheme to allow a user to access the devices deployed in their smart agricultural field using their personal mobile devices. This scheme is based on elliptic curve cryptography, one-way hashing, and biometric fuzzy extraction concepts. A willing user registers with the system through a Trusted Registration Authority (TRA). Similarly, all the smart

devices are registered with the TRA. In the authentication scheme, the user generates two secrets, one only known to the user and the other shared with a trusted controller node. The smart device also computes two secrets but only shares them after hiding them using hashing and bitwise exclusive-OR operations. The resultant secret session key is made of these the first two secrets concatenated with the product of the last two secrets as a Diffie-Hellman key. It also consists of a mobile device revocation phase, dynamic smart addition phase, and user biometric and password change phases. The scheme is analyzed using the formal ROR model, informal analysis, formal security verification using AVISPA, and cost computations using the MIRACL library.

The second contribution of this thesis is provided in **Chapter** 5. It proposes a private blockchain-based mutual authentication scheme, AKMS-AgriIoT, to allow a device in a flying zone of a field to send its sensed data from the agri-zone to the ground station server via a drone that is monitoring the zone. The ground station server forwards the data to the blockchain center consisting of cloud servers. This certificate-based scheme uses elliptic curve cryptography, t-degree symmetric bivariate polynomials, one-way hashing, and private blockchain technology. Authentication is required in two phases. In the first phase, the IoT smart devices and the respective drone in the zone authenticate each other and establish a session key, followed by the second phase in which the drone and the ground station server perform authentication leading to an established session key. The established session keys in the authentication process are used to encrypt and transfer the confidential sensor data to the ground station server. The GSS, which has sufficient computational resources, creates transactions from the received sensor data and encrypts them with its public key. These encrypted transactions are securely sent to the cloud server, which creates a block and adds it to the blockchain after a successful consensus from its peer cloud servers using the PBFT-based proposed consensus algorithm. The leader for this consensus is selected in a round-robin fashion. This scheme is analyzed using the formal ROR model, informal analysis, formal security verification using AVISPA, and cost computations using the MIRACL library. A blockchain simulation using Node.js has also been provided.

The third contribution of this thesis is provided in **Chapter** 6. It proposes a mutual authentication scheme, SCBAS-SF, that allows the IoT smart devices from various farm zones in an agricultural field to relay sensor data to a blockchain center via gateway servers and edge servers. It uses elliptic curve cryptography, one-way hashing, edge computing, hybrid blockchain technology, and smart contracts. The gateway servers, edge

servers, and cloud servers are registered offline. The authentication is accomplished in two phases. The first phase allows authentication between any two IoT smart devices with a session key agreement at the end. The second phase authenticates the IoT smart devices and the gateway servers with the establishment of a session key. The established session keys are used to encrypt and transfer sensor data from one device to the gateway server in a hop-by-hop manner. The gateway server creates transactions with signatures on the hash of transactions using ECDSA and sends them to the edge server. Transactions that contain sensitive data are encrypted as a whole transaction unit with the public key of the gateway node, while other transactions are unencrypted. A set of transactions to be forwarded to the edge server are encrypted with the public key of the edge server. The edge server verifies the transaction signatures, creates a partial block with its signature on it, and forwards it to the cloud server. The cloud server creates a full block and adds it to the blockchain after successful consensus using a smart contract. Each block consists of a mixture of some encrypted transactions and some other unencrypted transactions. This scheme is analyzed using the formal ROR model, informal analysis, formal security verification using AVISPA, and cost computations using the MIRACL library. A simulation of the presented blockchain model is shown using the hyperledger sawtooth.

The fourth and final contribution of this thesis is provided in **Chapter** 7. It proposes a mutual authentication scheme, AgroMobiBlock, on a smart agricultural system consisting of IoT smart sensor devices in each field divided into regions. Each field is under a fog system consisting of a P2P system of fog servers. Mobile farming vehicles roaming in the field regions collect sensor data from the smart devices and transfer it to their assigned fog servers. It uses elliptic curve cryptography, one-way hashing, an active hybrid blockchain technology, and fog computing. Critical credentials, including a mobile vehicle and fog server association key, are stored on the hybrid blockchain during the registration of the mobile vehicles and the fog servers. The authentication process is performed in two phases. The first phase authenticates the sensor nodes and the mobile vehicles to each other and establishes a session key to forward sensor data to mobile vehicles. The second phase of authentication between the mobile vehicles and fog servers retrieves the critical credentials from the hybrid blockchain and creates a session key from a long term secret from a mobile device encrypted using an association key, a private hash from a mobile device, a private hash from fog server, a long term secret from the fog server encrypted using the association key and a Diffie-Hellman type key. The P2P fog system maintains a hybrid blockchain consisting of either an AuthCred block to store the authentication

credentials stored during registration and retrieved during authentication, or a SensorData block, to store sensitive sensor data. During registration, transactions are created by the Trusted Registration Authority (TRA) while stored and retrieved as an AuthCred block by the fog servers. After authentication and key agreement are complete, the fog servers create transactions from the sensor data, encrypt each transaction with its own public key, concatenate encrypted transactions, and hashes them. A SensorData block is created out of this encrypted hashed set of transactions and added to the blockchain using the proposed PBFT-based consensus algorithm. Each block contains either only semi-encrypted transactions if it is an AuthCred block or only fully encrypted transactions if it is a SensorData block. No block can contain both semi-encrypted and encrypted transactions. This scheme is analyzed using the formal ROR model, informal analysis, formal security verification using AVISPA, and cost computations using the MIRACL library. A blockchain simulation using Node.js has also been provided.

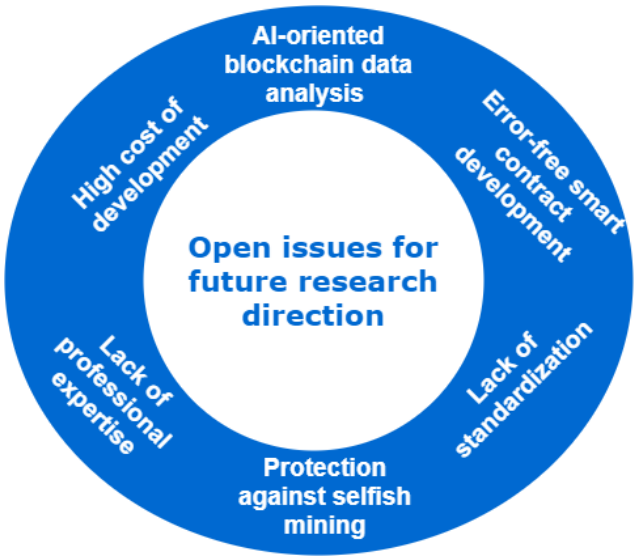

Figure 8.1: Open issues and challenges in blockchain-based smart sensing agriculture

8.2 Future research, open issues and challenges

In this section, we discuss many open issues and challenges that show the future directions to encourage active researchers to work in the blockchain-based smart sensing agriculture environment as represented in Figure 8.1.

- **AI-based prediction systems:** The development of security schemes can be enhanced using other fields, such as Artificial Intelligence (AI), Big data analytics, and Machine Learning (ML) [159, 207]. AI/ML can be used to analyze the data of the IoT device sensors stored in the blockchain. Security schemes for smart agriculture with blockchain can be furthered towards including AI/ML-based analysis by amalgamating the proposed architecture with the one given in Singh *et al.* [289]. Further research can be conducted to see how AI/ML can be used in data analysis and security schemes. Big data can be used along with blockchain technology to handle the ever-increasing amount of data collected from IoT smart devices. Thus, ML can also be used to develop learning modes that allow the system to upgrade its security system to resist new attacks for which the scheme was not initially designed.
- **Error-free development of smart contracts:** Smart contracts have the capability of automating the processes inside the blockchain. Any error or bug in a smart contract can lead to erroneous data being recorded in the blockchain. Since a blockchain can be used as a backbone for a number of applications, such bugs in smart contracts end in disastrous implications.
- **Lack of standardization:** Currently, no standards exist for the operation of a blockchain. It can lead to major incompatibility among organizations with regard to governance and interoperability.
- **High cost of development:** Adding a block to a blockchain is an expensive operation. It takes about USD 550 to add work to the blockchain. It is imperative to find methods to reduce this cost significantly.
- **Protection against selfish mining:** In a blockchain with a small number of nodes, it is crucial to ensure that computational resources are not piled by a single miner node or a small group of miner nodes in excess. Such a situation can lead to the successful tampering or reversing of the blockchain. There has been no instance of such an attack until now, but current systems are not equipped to deal with such circumstances in case they occur in the near future.
- **Physically secure authentication in smart agriculture:** A "Physically Unclonable Function (PUF)" is a special kind of one-way function with challenge-response mapping using a physical microstructure that is unique to every device in an IoT network. Such unique traits of devices align well with the security requirements needed for achieving authentication, untraceability, and access control.

The properties of an ideal PUF are listed below.

- The output obtained from a PUF is highly dependent on the physical system to which it belongs.
- Evaluation and construction of the PUF is easy.
- The output obtained from a PUF is highly unpredictable and hence can be a good representation of a random function.
- PUF is unclonable.

The concept of PUF in smart agriculture is by large unexplored and can find significant application in the design of authentication protocols.

Appendix A

AVISPA

AVISPA [9] is a simulation tool that allows to simulate the security protocols using "High-Level Protocol Specification Language (HLPSL)" and determine whether a security protocol is "safe", "unsafe" or "inconclusive". The simulation uses "SPAN, the Security Protocol ANimator for AVISPA" [22]. Presently, since the AVISPA implements only the Dolev-Yao (DY) threat model, it can detect both "replay and man-in-the-middle (MiTM)" attacks during communication by a passive/active adversary. A scheme is verified in three ways: a) "executability checking on non-trivial HLPSL specifications", b) "replay attack checking", and c) "DY model checking". If a protocol model does not complete its execution due to any modeling mistakes, no attack may be found by the backends as the attack state is never reached. In such cases, "executability check for non-trivial HLPSL specifications" is used as an important criterion for formal security verification under the AVISPA tool [322]. The backends search for a passive intruder and feed it with knowledge from honest agents to test against replay attack [9]. The DY model is checked by identifying if a "man-in-the-middle attack" can be launched with the identified intruder.

The architecture of AVISPA is shown in Figure A.1 and described in detail in the following sections.

A.1 Coding language and file formats

(1) *High Level Protocol Specification Language (HLPSL)*: A designed protocol requires to implement using a language based on temporal logic, called "High-Level Protocol

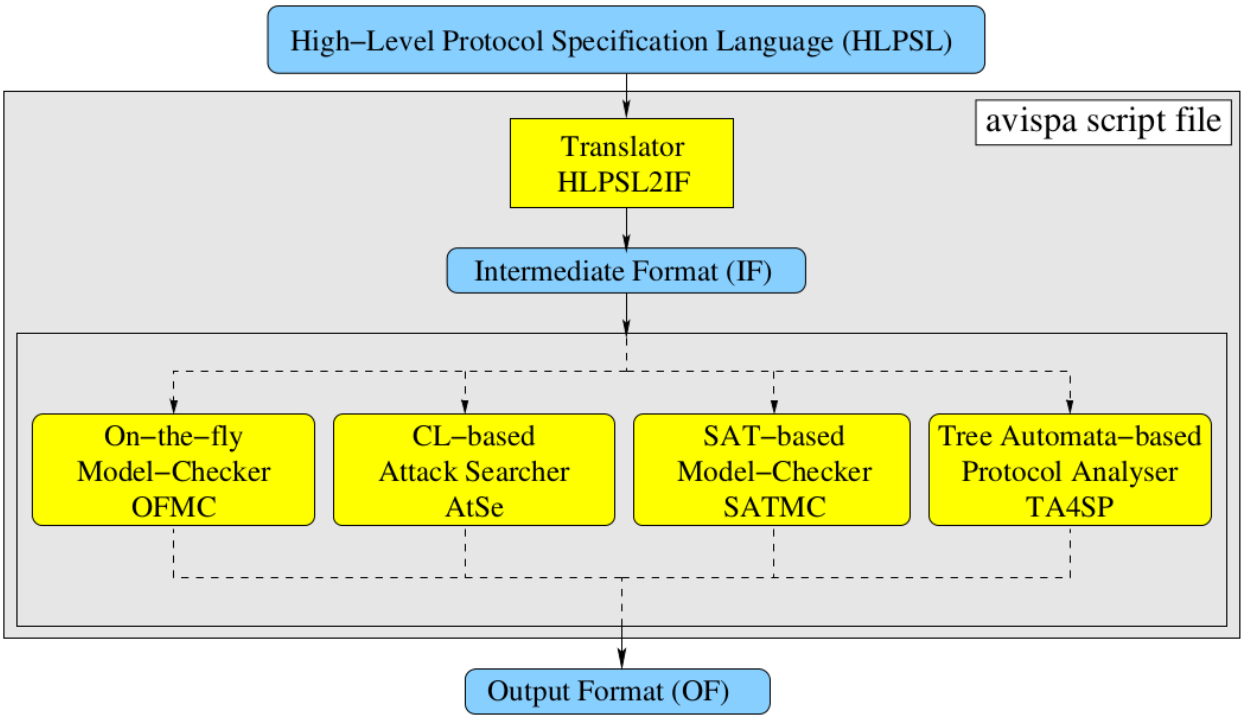

Figure A.1: Architecture of AVISPA tool

Specification Language (HLPSL)", which is used to simulate a protocol and test its safety against replay and MiTM attacks. The HLPSL has the ability to represent a protocol in terms of its various roles that are arranged in a hierarchy, with each role consisting of parameters, states, and transitions between states. It also allows the creation of a composition of roles. The details of the information known to an intruder, the goal of the protocol, and the environment with the composition of the roles need to be specified. The HLPSL code is first converted into the "Intermediate Format (IF)" that is read directly by one of the four available backends using the HLPSL2IF translator. HLPSL code uses temporal logic consisting of basic roles with transitions among their states apart from two mandatory roles for session and environment.

(2) *Intermediate Format (IF)*: The intermediate format is a lower-level specification obtained after translation from the HLPSL. This format is flexible to be used with any of the existing backends and with possible new backends that may be designed in the future.

(3) *Output Format (OF)*: The result of the analysis on IF consists of the following:

- Summary: It specifies if a scheme if "safe", "unsafe" or "inconclusive".
- Details: It specifies details of the session and the model used.
- Protocol: It specifies the location of the IF file that was used by the backend.
- Goal: It specifies if the same goal from the HLPSL is used for the analysis.
- Backend: It specifies the name of the backend that performed the analysis.
- Statistics: It specifies the number of states that were analyzed and reached along with the time taken for translation and computational analysis.

A.2 Components of AVISPA

(1) **Translator (HLPSL2IF):** The HLPSL language is a high-level language that needs to be converted to a low-level specification of IF. This translation is called automatically and is invisible to the user.

(2) **AVISPA Backends:** The purpose of the AVISPA backends is to analyze the IF specification in different ways. The AVISPA tool consists of four backends whose analysis methods complement each other, regardless of having common techniques and may return different results.

(a) *On The Fly Model Checker (OFMC)*: It constructs an on-demand infinite tree using symbols for the state space.

- Usage: OFMC can be used to prove a protocol to be accurate and to detect attacks fast for an unbounded number of messages from an intruder in a bounded number of sessions.
- It is based on algebraic theory applied to the terms inside messages.

(b) *Constraint Logic-based Attack Searcher (CL-AtSe)*: It creates a set of constraints from the transition relations specified in the IF for any security protocol. Every step in the protocol is modeled as a constraint on the knowledge that an adversary has.

- Automatic: This backend performs the translation and checking processes automatically without the need for any external tool.
- CL-AtSe is deterministic: The specified protocol must end in a bounded number of steps. If loops are involved, they should include the maximum iterations allowed.

- Simplified processing: CL-AtSe aims to keep the number of steps to be verified as low as possible by marking the steps for either immediate or deferred execution. This results in reduced interleaving of protocol steps.
- Operators: CL-AtSe supports the XOR operator and exponential operator.

(c) *SAT-based Model Checker (SATMC)*: It takes the transition relations from the IF, an initial state, and the set of states which violate security properties to compute a propositional formula. An SAT solver takes the formula to produce an attack model.

- SATMC is modular, flexible, and efficient.
- SATMC can discover attacks and verify if a protocol exhibits required security properties for a bounded number of sessions.

(d) *Tree Automate based on Automatic Approximations for the Analysis of Security Protocols (TA4SP)*: For an unbounded number of sessions, TA4SP rewrites tree languages to build an approximation of the intruder knowledge. This may result in over-approximation that shows a protocol to satisfy a security property for a given initial state and abstractions. Alternatively, it may result in under-approximation, which shows a protocol violating a security property.

Appendix B

MIRACL

The "Multiprecision Integer and Rational Arithmetic Cryptographic Library (MIRACL)" [38] is "a C/C++ based programming software library that has been already recognized by the cryptographers as the gold standard open-source SDK for elliptic curve cryptography (ECC)". Under this section, we evaluate various cryptographic primitives using the broadly-accepted MIRACL for their execution time.

We utilize the symbols T_{exp}, T_{ecm}, T_{eca}, T_{senc}/T_{sdec}, T_h, T_{mul}, T_{add} and T_{bp} to signify the time required for "modular exponentiation", "elliptic curve point (scalar) multiplication", "elliptic curve point addition", "symmetric key encryption/decryption using the Advanced Encryption Standard (AES-128) [8]", "one-way hash function using SHA-256 hashing algorithm [232]", "modular multiplication over $GF(q)$", "modular addition over $GF(q)$", and bilinear pairing respectively. The elliptic curve point addition and multiplication are carried out on a non-singular elliptic curve of the type: "$y^2 = x^3 + ux + v \pmod q$" such that $4u^3 + 27v^2 \neq 0 \pmod q$.

We consider the following two scenarios for experiments using MIRACL:

- **Scenario 1:** In this case, we consider the platform for a server as follows: "Ubuntu 18.04.4 LTS, with memory: 7.7 GiB, processor: Intel Core i7-8565U CPU @ 1.80GHz × 8, OS type: 64-bit and disk: 966.1 GB". The experiments for each cryptographic primitive are performed for 100 runs. From these 100 runs, we recorded the "maximum, minimum, and average run-time in milliseconds for each cryptographic primitive". In Table B.1, we have then tabulated the experimental results.
- **Scenario 2:** In this case, we consider the platform for a smart device/drone as

Table B.1: Execution time (in milliseconds) of cryptographic primitives using MIRACL on a server

Primitive	Max. time (ms)	Min. time (ms)	Average time (ms)
T_{ecm}	2.998	0.284	0.674
T_{eca}	0.002	0.001	0.002
T_{exp}	0.248	0.046	0.072
T_h	0.149	0.024	0.055
T_{mul}	0.007	0.001	0.002
T_{add}	0.003	0.001	0.001
T_{senc}	0.003	0.001	0.001
T_{sdec}	0.002	0.001	0.001

Table B.2: Execution time (in milliseconds) of cryptographic primitives using MIRACL on a Raspberry PI 3

Primitive	Min. time (ms)	Max. time (ms)	Average time (ms)
T_{ecm}	2.206	4.532	2.288
T_{eca}	0.015	0.021	0.016
T_{exp}	0.178	0.493	0.228
T_h	0.274	0.643	0.309
T_{mul}	0.009	0.016	0.011
T_{add}	0.008	0.013	0.010
T_{senc}	0.017	0.038	0.018
T_{sdec}	0.009	0.054	0.014

follows: "Raspberry PI 3 B+ Rev 1.3, with CPU: 64-bit, Processor: 1.4 GHz Quad-core, 4 cores, Memory (RAM): 1GB, and OS: Ubuntu 20.04 LTS, 64-bit [21]". Similar to Scenario 1, the experiments for each cryptographic primitives are also performed for 100 runs. From these 100 runs, we recorded the "maximum, minimum, and average run-time in milliseconds for each cryptographic primitive". Next, we have shown the experimental results in Table B.2.

Table B.3 summarizes the average time for the execution of cryptographic schemes on

Table B.3: Average execution time (in milliseconds)

Primitive	Average time on Raspberry PI 3 (ms)	Average time on server (ms)
T_h	0.309	0.055
T_{exp}	0.228	0.072
T_{ecm}	2.288	0.674
T_{eca}	0.016	0.002
T_{senc}	0.018	0.001
T_{sdec}	0.014	0.001
T_{bp}	32.084	4.603

the server and on Raspberry PI 3.

Appendix C

Node.js

Node.js [39, 40, 41] is a single-threaded, open-source, cross-platform runtime environment to develop servers and networking applications. The V8 JavaScript runtime engine is used to run Node.js. It is event-driven with a non-blocking I/O architecture. Node.js consists of Node Package Management (NPM), the largest open-source library with a command-line utility. Node.js can be used to create REST API servers, real-time chatting software, data streaming applications, IoT applications, and web applications.

The architecture of Node.js is shown in Figure C.1.

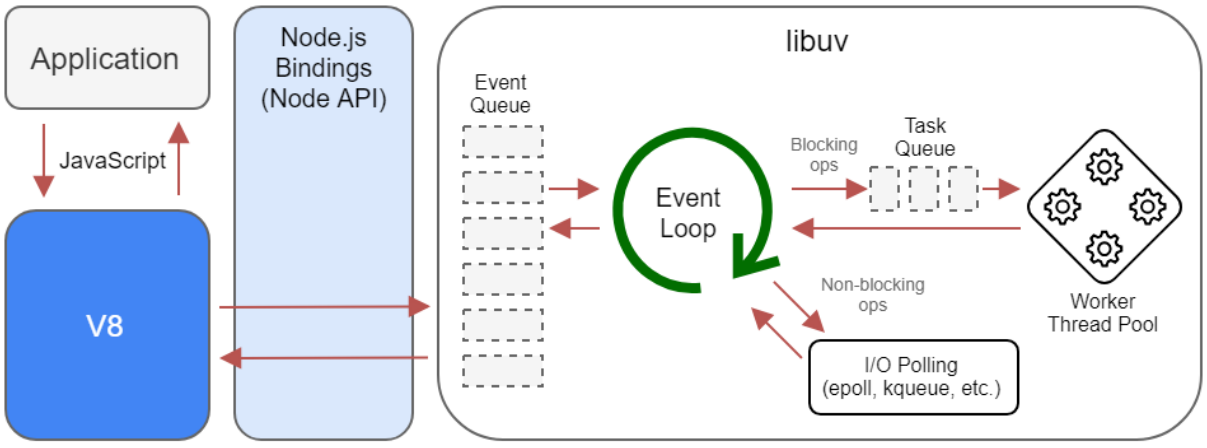

Figure C.1: Architecture of Node.js

Node.js contains a limited pool of auxiliary threads called worker group. A single-threaded Event Loop component of Node.js waits for requests. For every request received, Node.js

places it in a request queue. The Event Loop picks a request from the queue and checks if it requires blocking I/O. If so, it assigns a thread from the worker group. If not, the request is processed immediately, and a response is sent. This way, Node.js supports multiple concurrent requests. Node.js is ideal for real-time applications as it uses very few threads and other resources. It cannot be used for data-intensive tasks.

Appendix D

Hyperledger sawtooth

Hyperledger Sawtooth is an open-source blockchain platform from The Linux Foundation for permissioned and permissionless blockchains. It allows all the peer systems to access all the available transactions. It follows the regular Order-Execute-Commit flow of events. The processing, ordering, and delivery of transactions are achieved using a hyperledger validator. It supports "Proof-of-Elapsed-Time (PoET)" and "Practical Byzantine Fault Tolerance (PBFT)" consensus algorithms based on the Byzantine Fault consensus on specialized hardware.

D.1 Hyperledger sawtooth architecture

Hyperledger sawtooth [172] offers a flexible and modular architecture that separates the core system from the application domain, and smart contracts can specify the business rules for applications without needing to know the underlying design of the core system. The nodes participating in the consensus process of a Sawtooth chain are called "validator" nodes. The Sawtooth framework facilitates modeling smart contracts as a state machine, often called a "transaction processor". A transaction processor is a pluggable module that gets registered with the validator node in a Sawtooth chain. After passing through the distributed log, transactions are routed to the appropriate transaction processor by each validator node. The Sawtooth platform is agnostic to the transaction processing language and supports various high-level programming languages, such as Java and Python. The architecture for the hyperledger sawtooth is shown in Figure D.1.

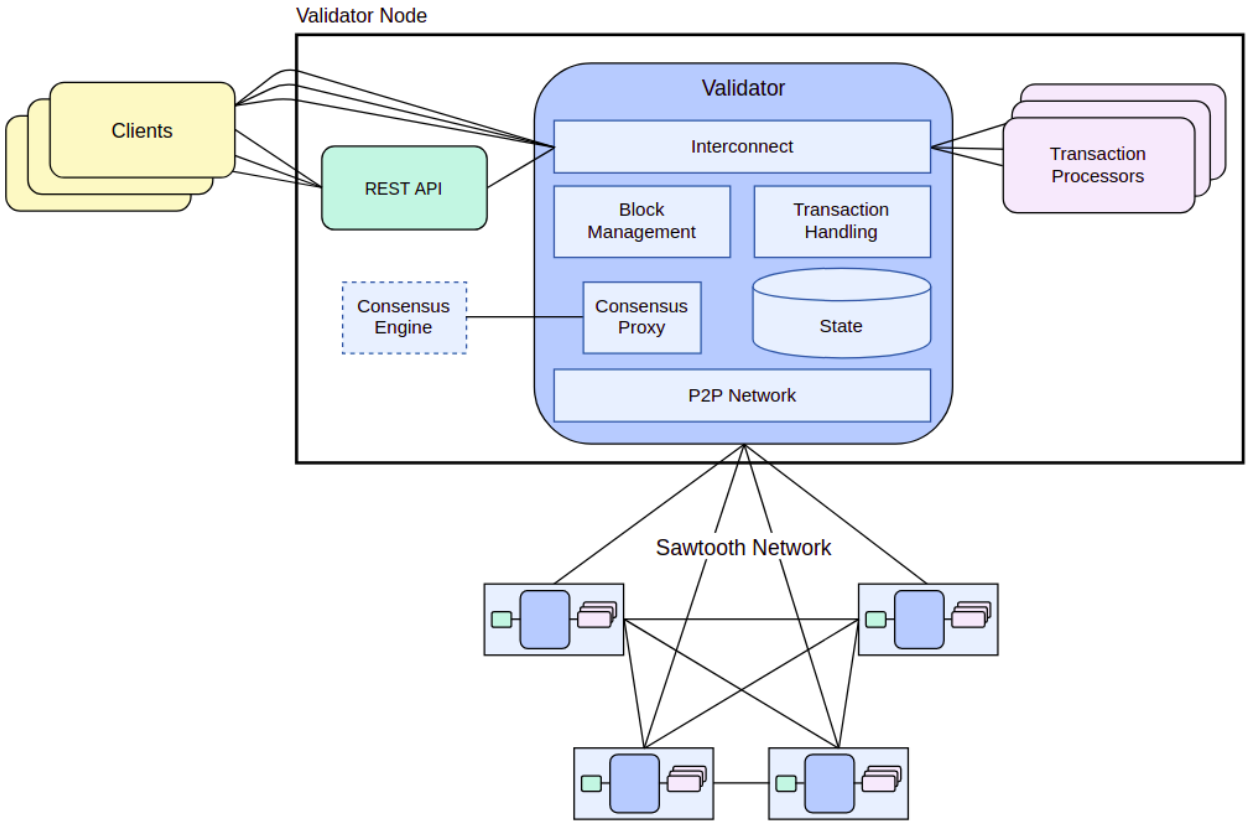

Figure D.1: Architecture of hyperledger sawtooth

D.2 Features of hyperledger sawtooth architecture

The hyperledger sawtooth architecture consists of the following features:

- *Global state:* All the participating nodes agree on a global state using a Byzantine consensus. The state is represented as an instance of the Merkle-Radix Tree on every validator node. Defining the global state includes defining and encoding the unique addresses to all leaf nodes and a deterministic serialization/de-serialization mechanism.
- *Transactions and batches:* Transactions are used to modify the global state of the distributed ledger. Transactions are executed in collections called batches, where all the transactions in a batch are in the same state. Either all the transactions move to a state or none of them. A set of possible transactions that can be added is called a transaction family. The “Chain Controller” and “ Block Publisher” components compute the state changes and the associated Merkle Tree hashes using schedulers

that allow either serial or parallel execution of a transaction. The "Executor" component sends the transactions to the transaction processors in the order defined by the schedulers.

- *Journal:* The subcomponents of the validator nodes collectively handle the batches of transactions, and the proposed blocks are collectively called a journal.
- *Sawtooth network:* The Sawtooth architecture consists of a self-contained network layer that manages communication between the validator nodes. It has the responsibility of connectivity, discovering neighboring peers, and message passing.
- *Event:* Any important occurrence related to the distributed ledger or the transactions is called an event. Examples of these include the addition of a new block and the execution of a transaction.
- *Transaction receipts:* Any information about the execution status of a transaction that need not be stored on the ledger but needs to be provided to the clients are stored as transaction receipts.

Bibliography

[1] Decade's Worst Food Shortage in North Korea. https://timesofindia.indiatimes.com/world/rest-of-world/north-korea-says-its-facing-worst-food-shortage-in-decade/articleshow/84404761.cms.

[2] DPOS Consensus Algorithm - The Missing White Paper. https://steemit.com/dpos/@dantheman/dpos-consensus-algorithm-this-missing-white-paper.

[3] Economic Crisis in Sri Lanka. https://www.thehindu.com/business/explained-what-caused-the-sri-lankan-economic-crisis/article36314148.ece.

[4] Keccak Specifications Summary. https://keccak.team/keccak_specs_summary.html.

[5] Local food systems and COVID-19; A glimpse on India's responses. http://www.fao.org/in-action/food-for-cities-programme/news/detail/en/c/1272232/.

[6] Lockdown and early floods: Double trouble for Assam farmers. https://indianexpress.com/article/india/lockdown-and-early-floods-double-trouble-for-assam-farmers-6511297/.

[7] National e-Governance Plan in Agriculture (NeGP-A).

[8] Advanced Encryption Standard. FIPS PUB 197, National Institute of Standards and Technology (NIST), U.S. Department of Commerce, November 2001., 2001. http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf.

[9] Automated Validation of Internet Security Protocols and Applications, 2006. http://www.avispa-project.org/.

[10] Food Wastage Footprint - Full Cost Accounting, 2014. http://www.fao.org/3/a-i3991e.pdf.

[11] Intel Software Guard Extensions Programming Reference, 2014. https://software.intel.com/sites/default/files/managed/48/88/329298-002.pdf.

[12] Slimcoin : A Peer-to-Peer Crypto-Currency with Proof-of-Burn - Mining without Powerful Hardware, 2014. https://github.com/slimcoin-project/slimcoin-project.github.io/raw/master/whitepaperSLM.pdf.

[13] The XRP Ledger, 2014. https://xrpl.org/consensus-principles-and-rules.html.

[14] e-Agriculture in Action: Blockchain for Agriculture - Opportunities and Challenges, 2019. http://www.fao.org/3/CA2906EN/ca2906en.pdf.

[15] KhethiNext, 2019. www.khethinext.com.
[16] The State of Food and Agriculture, 2019. http://www.fao.org/3/ca6030en/ca6030en.pdf.
[17] World Population Prospects, 2019. United Nations, Department of Economic and Social Affairs, Population Division.
[18] World Population Prospects 2019: Highlights, 2019. https://population.un.org/wpp/Publications/Files/WPP2019_Highlights.pdf.
[19] Introducting Zero-G Network - Sigfox, March 2020. https://www.sigfox.com/sites/default/files/og-guide/Sigfox%20-%20Introducing%20OG_A5_March2020_4.pdf.
[20] *Modelling, Control and Simulation of an Unmanned Ground Vehicle for Agriculture 4.0*. PhD thesis, Politecnico di Torino, 2020.
[21] Raspberry Pi 3 Model B+, 2020. https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/.
[22] SPAN, the Security Protocol ANimator for AVISPA, 2020. http://www.avispa-project.org/.
[23] The essential commodities (amendment) ordinance bill, 2020, September 2020. http://egazette.nic.in/WriteReadData/2020/219748.pdf.
[24] The farmers (empowerment and protection) agreement on price assurance and farm services ordinance bill, 2020, September 2020. http://www.egazette.nic.in/WriteReadData/2020/219750.pdf.
[25] The farmers' produce trade and commerce (promotion and facilitation) bill, 2020, September 2020. http://www.egazette.nic.in/WriteReadData/2020/219745.pdf.
[26] Blynk Unified Platform, 2021. https://blynk.io/.
[27] FIWARE Cygnus - Tuning Tips for Increasing the Performance, 2021. https://fiware-orion.readthedocs.io/en/master/.
[28] FIWARE Cygnus - Tuning Tips for Increasing the Performance, 2021. https://fiware-cygnus.readthedocs.io/en/latest/.
[29] FIWARE: The Open Source Platform for Our Smart Digital Future, 2021. https://www.fiware.org/.
[30] Indian Agriculture and Allied Industries Report, June 2021. https://www.ibef.org/industry/agriculture-india.aspx.
[31] Infosys Precision Crop Management Testbed, 2021. https://www.infosys.com/industries/agriculture/industry-offerings/precision-farming.html.
[32] LoRa Alliance, 2021. https://lora-alliance.org/.
[33] Matlab, 2021. https://www.mathworks.com/products/matlab.html.
[34] MongoDB, 2021. https://www.mongodb.com/.
[35] Smart Farming with IoT and Cloud in Malaysia, 2021. https://techwireasia.com/2021/08/smart-farming-with-iot-and-cloud-in-malaysia/.
[36] The Internet of Things for Precision Agriculture, an NSF Engineering Research Center, 2021. https://iot4ag.us/products/.

[37] ThingSpeak for Smart Farming, 2021. https://thingspeak.com/pages/smart_farming.

[38] MIRACL Cryptographic SDK: Multiprecision Integer and Rational Arithmetic Cryptographic Library, 2022. https://github.com/miracl/MIRACL.

[39] Node.js, 2022. https://kinsta.com/knowledgebase/what-is-node-js/.

[40] Node.js, 2022. https://nodejs.org/en/.

[41] Node.js, 2022. https://nodejs.dev/learn.AccessedonApril2022.

[42] The war in Ukraine is exposing gaps in the world's food-systems research. *Nature*, 604(7905):217–218, 2022. https://www.nature.com/articles/d41586-022-00994-8.

[43] M. Abdalla, P. A. Fouque, and D. Pointcheval. Password-based authenticated key exchange in the three-party setting. In *8th International Workshop on Theory and Practice in Public Key Cryptography (PKC'05), Lecture Notes in Computer Science*, volume 3386, pages 65–84, Les Diablerets, Switzerland, 2005.

[44] F. J. Abdullayeva. Advanced Persistent Threat attack detection method in cloud computing based on autoencoder and softmax regression algorithm. *Array*, 10:100067, 2021.

[45] S. Aggarwal, R. Chaudhary, G. S. Aujla, N. Kumar, K.-K. R. Choo, and A. Y. Zomaya. Blockchain for Smart Communities: Applications, challenges and opportunities. *Journal of Network and Computer Applications*, 144:13 – 48, 2019.

[46] R. Z. Ahmed and R. C. Biradar. Data aggregation for pest identification in coffee plantations using WSN: A hybrid model. In *International Conference on Computing and Network Communications (CoCoNet)*, pages 139–146, 2015.

[47] R. Z. Ahmed and R. C. Biradar. Redundancy aware data aggregation for pest control in coffee plantation using wireless sensor networks. In *2nd International Conference on Signal Processing and Integrated Networks (SPIN)*, pages 984–989, 2015.

[48] O. Ahumada and J. R. Villalobos. Application of planning models in the agri-food supply chain: A review. *European Journal of Operational Research*, 196(1):1–20, 2009.

[49] S. V. Akram, P. K. Malik, R. Singh, G. Anita, and S. Tanwar. Adoption of blockchain technology in various realms: Opportunities and challenges. *Security and Privacy*, 3(5):e109, 2020.

[50] G. Alfian, J. Rhee, H. Ahn, J. Lee, U. Farooq, M. F. Ijaz, and M. A. Syaekhoni. Integration of RFID, wireless sensor networks, and data mining in an e-pedigree food traceability system. *Journal of Food Engineering*, 212:65–75, 2017.

[51] G. Alfian, M. Syafrudin, U. Farooq, M. R. Ma'arif, M. A. Syaekhoni, N. L. Fitriyani, J. Lee, and J. Rhee. Improving efficiency of RFID-based traceability system for perishable food by utilizing IoT sensors and machine learning model. *Food Control*, 110:107016, 2020.

[52] R. Ali, A. K. Pal, S. Kumari, M. Karuppiah, and M. Conti. A secure user authentication and key-agreement scheme using wireless sensor networks for agriculture monitoring. *Future Generation Computer Systems*, 84:200–215, 2018.

[53] R. Almadhoun, M. Kadadha, M. Alhemeiri, M. Alshehhi, and K. Salah. A User Authentication Scheme of IoT Devices using Blockchain-Enabled Fog Nodes. In *15th IEEE/ACS*

International Conference on Computer Systems and Applications (AICCSA), pages 1–8, Aqaba, Jordan, 2018.

[54] S. Alyahya, W. U. Khan, S. Ahmed, S. N. K. Marwat, and S. Habib. Cyber Secure Framework for Smart Agriculture: Robust and Tamper-Resistant Authentication Scheme for IoT Devices. *Electronics*, 11(6), 2022.

[55] A. F.-X. Ametepe, S. A. R. M. Ahouandjinou, and E. C. Ezin. Secure Encryption by Combining Asymmetric and Symmetric Cryptographic Method for Data Collection WSN in smart Agriculture. In *IEEE International Smart Cities Conference (ISC2)*, pages 93–99, 2019.

[56] T. Anand, S. Sinha, M. Mandal, V. Chamola, and F. R. Yu. Agrisegnet: Deep aerial semantic segmentation framework for iot-assisted precision agriculture. *IEEE Sensors Journal*, 21(16):17581–17590, 2021.

[57] N. Andola, S. Prakash, R. Gahlot, S. Venkatesan, and S. Verma. An enhanced smart card and dynamic ID based remote multi-server user authentication scheme. *Cluster Computing*, pages 1–19, 2022.

[58] J. Arshad, M. A. B. Siddique, Z. Zulfiqar, A. Khokhar, S. Salim, T. Younas, A. U. Rehman, and A. Asad. A Novel Remote User Authentication Scheme by using Private Blockchain-Based Secure Access Control for Agriculture Monitoring. In *International Conference on Engineering and Emerging Technologies (ICEET)*, pages 1–9, Lahore, Pakistan, 2020.

[59] H. F. Atlam, A. Alenezi, M. O. Alassafi, and G. Wills. Blockchain with Internet of Things: benefits, challenges, and future directions. *International Journal of Intelligent Systems and Applications*, 10(6):40–48, June 2018.

[60] L. Atzori, A. Iera, and G. Morabito. The Internet of Things: A survey. *Computer Networks*, 54(15):2787 – 2805, 2010.

[61] M. H. Au, K. Liang, J. K. Liu, R. Lu, and J. Ning. Privacy-preserving personal data operation on mobile cloud - Chances and challenges over advanced persistent threat. *Future Generation Computer Systems*, 79:337–349, 2018.

[62] M. Bacco, P. Barsocchi, E. Ferro, A. Gotta, and M. Ruggeri. The Digitisation of Agriculture: a Survey of Research Activities on Smart Farming. *Array*, 3-4:100009, 2019.

[63] R. Badia-Melis, P. Mishra, and L. Ruiz-García. Food traceability: New trends and recent advances. A review. *Food Control*, 57:393–401, 2015.

[64] Q. Bai, X. Zhou, X. Wang, Y. Xu, X. Wang, and Q. Kong. A Deep Dive Into Blockchain Selfish Mining. In *IEEE International Conference on Communications (ICC)*, pages 1–6, 2019.

[65] S. Banerjee, V. Odelu, A. K. Das, S. Chattopadhyay, N. Kumar, Y. Park, and S. Tanwar. Design of an Anonymity-Preserving Group Formation Based Authentication Protocol in Global Mobility Networks. *IEEE Access*, 6:20673–20693, 2018.

[66] E. Barker. Recommendation for Key Management. Special Publication 800-57 Part 1 Rev. 4, NIST, 01/2016. Accessed on April2022., 2016. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r4.pdf.

[67] L. Barreto and A. Amaral. Smart Farming: Cyber Security Challenges. In *2018 International Conference on Intelligent Systems (IS)*, pages 870–876, 2018.

[68] D. Basin, C. Cremers, and S. Meier. Provably repairing the ISO/IEC 9798 standard for entity authentication. *Journal of Computer Security*, 21(6):817–846, 2013.

[69] R. Belguechi, C. Rosenberger, and S. Ait-Aoudia. Biohashing for securing minutiae template. In *20th International Conference on Pattern Recognition (ICPR'10)*, pages 1168–1171, Istanbul, Turkey, 2010.

[70] M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated Key Exchange Secure against Dictionary Attacks. In B. Preneel, editor, *Advances in Cryptology — EUROCRYPT 2000*, pages 139–155, Bruges, Belgium, 2000.

[71] I. Bentov, C. Lee, A. Mizrahi, and M. Rosenfeld. Proof of Activity: Extending Bitcoin's Proof of Work via Proof of Stake. *SIGMETRICS Perform. Eval. Rev.*, 42(3):34–37, 2014.

[72] I. Bentov, R. Pass, and E. Shi. Snow White: Provably Secure Proofs of Stake. *IACR Cryptology ePrint Archive*, 2016(919), 2016.

[73] B. Bera, D. Chattaraj, and A. K. Das. Designing secure blockchain-based access control scheme in IoT-enabled Internet of Drones deployment. *Computer Communications*, 153:229 – 249, 2020.

[74] B. Bera, S. Saha, A. K. Das, N. Kumar, P. Lorenz, and M. Alazab. Blockchain-Envisioned Secure Data Delivery and Collection Scheme for 5G-Based IoT-Enabled Internet of Drones Environment. *IEEE Transactions on Vehicular Technology*, 69(8):9097–9111, 2020.

[75] E. Bertino, N. Shang, and S. S. Wagstaff Jr. An Efficient Time-Bound Hierarchical Key Management Scheme for Secure Broadcasting. *IEEE Transactions on Dependable and Secure Computing*, 5(2):65–70, 2008.

[76] J. Bhatia, Y. Modi, S. Tanwar, and M. Bhavsar. Software defined vehicular networks: A comprehensive review. *International Journal of Communication Systems*, 32(12):e4005, 2019.

[77] P. Bhattacharya, S. Tanwar, U. Bodke, S. Tyagi, and N. Kumar. BinDaaS: Blockchain-Based Deep-Learning as-a-Service in Healthcare 4.0 Applications. *IEEE Transactions on Network Science and Engineering*, 2019. DOI: 10.1109/TNSE.2019.2961932.

[78] M. N. M. Bhutta and M. Ahmad. Secure identification, traceability and real-time tracking of agricultural food supply during transportation using internet of things. *IEEE Access*, 9:65660–65675, 2021.

[79] A. Biryukov, D. Khovratovich, and I. Pustogarov. Deanonymisation of Clients in Bitcoin P2P Network. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 15–29, 2014.

[80] C. Blundo, A. De Santis, A. Herzberg, S. Kutten, U. Vaccaro, and M. Yung. Perfectly Secure Key Distribution for Dynamic Conferences. *Information and Computation*, 146(1):1–23, 1998.

[81] U. Bodkhe, P. Bhattacharya, S. Tanwar, S. Tyagi, N. Kumar, and M. S. Obaidat. BloHosT: Blockchain Enabled Smart Tourism and Hospitality Management. In *International Conference on Computer, Information and Telecommunication Systems (CITS)*, pages 1–5, Beijing, China, 2019.

[82] U. Bodkhe, D. Mehta, S. Tanwar, P. Bhattacharya, P. K. Singh, and W. Hong. A Survey on Decentralized Consensus Mechanisms for Cyber Physical Systems. *IEEE Access*, 8:54371–54401, 2020.

[83] U. Bodkhe, S. Tanwar, K. Parekh, P. Khanpara, S. Tyagi, N. Kumar, and M. Alazab. Blockchain for Industry 4.0: A Comprehensive Review. *IEEE Access*, 8:79764–79800, 2020.

[84] A. Boghossian, S. Linsky, A. Brown, P. Mutschler, B. Ulicny, L. Barrett, et al. Threats to precision agriculture. *US Department of Homeland Security, Washington, DC, USA, Tech. Rep. 20181003a*, 2018.

[85] D. Boneh, E. Shen, and B. Waters. Strongly Unforgeable Signatures Based on Computational Diffie-Hellman. In *International Workshop on Public Key Cryptography (PKC'06)*, pages 229–240, New York, USA, 2006.

[86] M. Bor, J. E. Vidler, and U. Roedig. LoRa for the Internet of Things. In *EWSN '16 Proceedings of the 2016 International Conference on Embedded Wireless Systems and Networks*, pages 361–366. Junction Publishing, AUT, 2016.

[87] T. Bosona and G. Gebresenbet. Food traceability as an integral part of logistics management in food and agricultural supply chain. *Food Control*, 33(1):32–48, 2013.

[88] A. Bothe, J. Bauer, and N. Aschenbruck. RFID-assisted Continuous user authentication for IoT-based smart farming. In *IEEE International Conference on RFID Technology and Applications (RFID-TA)*, pages 505–510, 2019.

[89] M. Bouam, C. Bouillaguet, C. Delaplace, and C. Noûs. Computational Records with Aging Hardware: Controlling Half the Output of SHA-256. Accessed on July 2022, 2021.

[90] A. D. Boursianis, M. S. Papadopoulou, P. Diamantoulakis, A. Liopa-Tsakalidi, P. Barouchas, G. Salahas, G. Karagiannidis, S. Wan, and S. K. Goudos. Internet of Things (IoT) and Agricultural Unmanned Aerial Vehicles (UAVs) in smart farming: A comprehensive review. *Internet of Things*, page 100187, 2020.

[91] C. Brewster, I. Roussaki, N. Kalatzis, K. Doolin, and K. Ellis. IoT in Agriculture: Designing a Europe-Wide Large-Scale Pilot. *IEEE Communications Magazine*, 55(9):26–33, 2017.

[92] M. Burrows, M. Abadi, and R. Needham. A logic of authentication. *ACM Transactions on Computer Systems*, 8(1):18–36, 1990.

[93] V. Buterin. Thoughts on UTXOs. https://medium.com/@ConsenSys/thoughts-on-utxo-by-vitalik-buterin-2bb782c67e53.

[94] R. Canetti and H. Krawczyk. Universally Composable Notions of Key Exchange and Secure Channels. In *International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT'02)*, pages 337–351, Amsterdam, The Netherlands, 2002.

[95] M. Castro and B. Liskov. Practical Byzantine Fault Tolerance and Proactive Recovery. *ACM Trans. Comput. Syst.*, 20(4):398–461, 2002.

[96] C.-J. Chae and H.-J. Cho. Enhanced secure device authentication algorithm in P2P-based smart farm system. *Peer-to-peer networking and applications*, 11(6):1230–1239, 2018.

[97] C. C. Chang and H. D. Le. A Provably Secure, Efficient and Flexible Authentication Scheme for Ad hoc Wireless Sensor Networks. *IEEE Transactions on Wireless Communications*, 15(1):357–366, 2016.

[98] C.-C. Chang and N.-T. Nguyen. An Untraceable Biometric-Based Multi-server Authenticated Key Agreement Protocol with Revocation. *Wireless Personal Communications*, 90(4):1695–1715, 2016.

[99] S. Chatterjee, A. K. Das, and J. K. Sing. A novel and efficient user access control scheme for wireless body area sensor networks. *Journal of King Saud University - Computer and Information Sciences*, 26(2):181–201, 2014.

[100] S. Chatterjee, A. K. Das, and J. K. Sing. An Enhanced Access Control Scheme in Wireless Sensor Networks. *Ad Hoc & Sensor Wireless Networks*, 21(1-2):121–149, 2014.

[101] A. Chauhan, O. P. Malviya, M. Verma, and T. S. Mor. Blockchain and Scalability. In *IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, pages 122–128, Lisbon, Portugal, 2018.

[102] M. Chen, T.-F. Lee, and J.-I. Pan. An Enhanced Lightweight Dynamic Pseudonym Identity Based Authentication and Key Agreement Scheme Using Wireless Sensor Networks for Agriculture Monitoring. *Sensors*, 19(5), 2019.

[103] W. Chen, Y. Lin, Y. Lin, R. Chen, J. Liao, F. Ng, Y. Chan, Y. Liu, C. Wang, C. Chiu, and T. Yen. AgriTalk: IoT for Precision Soil Farming of Turmeric Cultivation. *IEEE Internet of Things Journal*, 6(3):5209–5223, 2019.

[104] M. E. H. Chowdhury, A. Khandakar, S. Ahmed, F. Al-Khuzaei, J. Hamdalla, F. Haque, M. B. I. Reaz, A. Al Shafei, and N. Al-Emadi. Design, Construction and Testing of IoT Based Automated Indoor Vertical Hydroponics Farming Test-Bed in Qatar. *Sensors*, 20(19), 2020.

[105] K. Christidis and M. Devetsikiotis. Blockchains and smart contracts for the internet of things. *IEEE Access*, 4:2292–2303, 2016.

[106] S. S. L. Chukkapalli, A. Piplai, S. Mittal, M. Gupta, and A. Joshi. A Smart-Farming Ontology for Attribute Based Access Control. In *IEEE 6th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and*

Smart Computing, (HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS), pages 29–34, 2020.

[107] D. M. A. Cortez, A. M. Sison, and R. P. Medina. Cryptanalysis of the Modified SHA256. In *4th High Performance Computing and Cluster Technologies Conference (HPCCT'20); 3rd International Conference on Big Data and Artificial Intelligence (BDAI'20)*, pages 179 – 183, Qingdao, China, 2020.

[108] C. Costa, F. Antonucci, F. Pallottino, J. Aguzzi, D. Sarriá, and P. Menesatti. A review on agri-food supply chain traceability by means of RFID technology. *Food and bioprocess technology*, 6(2):353–366, 2013.

[109] S. Cox. Information technology: the global key to precision agriculture and sustainability. *Computers and Electronics in Agriculture*, 36(2):93–111, 2002.

[110] Z. Cui, F. XUE, S. Zhang, X. Cai, Y. Cao, W. Zhang, and J. Chen. A Hybrid BlockChain-Based Identity Authentication Scheme for Multi-WSN. *IEEE Transactions on Services Computing*, 13(2):241–251, 2020.

[111] K. Dandage, R. Badia-Melis, and L. Ruiz-García. Indian perspective in food traceability: A review. *Food Control*, 71:217–227, 2017.

[112] A. K. Das. An unconditionally secure key management scheme for large-scale heterogeneous wireless sensor networks. In *First International Communication Systems and Networks and Workshops (COMSNETS'09)*, pages 1–10, Bangalore, India, 2009.

[113] A. K. Das. A Secure User Anonymity-Preserving Three-Factor Remote User Authentication Scheme for the Telecare Medicine Information Systems. *Journal of Medical Systems*, 39(3):30, 2015.

[114] A. K. Das, S. Kumari, V. Odelu, X. Li, F. Wu, and X. Huang. Provably secure user authentication and key agreement scheme for wireless sensor networks. *Security and Communication Networks*, 9(16):3670–3687, 2016.

[115] A. K. Das, A. K. Sutrala, S. Kumari, V. Odelu, M. Wazid, and X. Li. An efficient multi-gateway-based three-factor user authentication and key agreement scheme in hierarchical wireless sensor networks. *Security and Communication Networks*, 9(13):2070–2092, 2016.

[116] A. K. Das, M. Wazid, N. Kumar, A. V. Vasilakos, and J. J. P. C. Rodrigues. Biometrics-Based Privacy-Preserving User Authentication Scheme for Cloud-Based Industrial Internet of Things Deployment. *IEEE Internet of Things Journal*, 5(6):4900–4913, 2018.

[117] A. K. Das, S. Zeadally, and D. He. Taxonomy and analysis of security protocols for Internet of Things. *Future Generation Computer Systems*, 89:110 – 125, 2018.

[118] B. Das and D. Bora. Determinants of Farm Productivity in Flood Prone area: A Study in Dhemaji District of Assam. *Indian Journal of Agricultural Research*, 54(1):83–88, 2020.

[119] T. K. Dasaklis, F. Casino, and C. Patsakis. Defining Granularity Levels for Supply Chain Traceability Based on IoT and Blockchain. In *International Conference on Omni-Layer Intelligent Systems*, pages 184–190, Crete, Greece, 2019.

[120] K. Demestichas, N. Peppes, and T. Alexakis. Survey on Security Threats in Agricultural IoT and Smart Farming. *Sensors*, 20(22), 2020.

[121] S. Dharmaraj. Maharashtra to adopt blockchain tech for governance. https://opengovasia.com/maharashtra-india-to-adopt-blockchain-tech-for-governance/.

[122] P. K. Dhillon and S. Kalra. A lightweight biometrics based remote user authentication scheme for IoT services. *Journal of Information Security and Applications*, 34:255 – 270, 2017.

[123] L. Ding, J. Wu, X. Zhang, J. Li, and J. Ma. Privacy Preserved Cyber-Physical Searching for Information-Centric Intelligent Agriculture. *IEEE Open Journal of the Computer Society*, 2:106–116, 2021.

[124] Y. Dodis, L. Reyzin, and A. Smith. Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data. In *International Conference on the Theory and Applications of Cryptographic Techniques (Eurocrypt'04)*, pages 523–540, Interlaken, Switzerland, 2004.

[125] D. Dolev and A. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983.

[126] J. R. Douceur. The Sybil Attack. In *Peer-to-Peer Systems*, pages 251–260, Berlin, Heidelberg, 2002.

[127] A. Dua, N. Kumar, A. K. Das, and W. Susilo. Secure message communication protocol among vehicles in smart city. *IEEE Transactions on Vehicular Technology*, 67(5):4359–4373, 2017.

[128] M. Duque-Acevedo, L. J. Belmonte-Urena, F. J. Cortes-Garcia, and F. Camacho-Ferre. Agricultural waste: Review of the evolution, approaches and perspectives on alternative uses. *Global Ecology and Conservation*, 22:e00902, 2020.

[129] S. Dziembowski, S. Faust, V. Kolmogorov, and K. Pietrzak. Proofs of space. In *Advances in Cryptology – CRYPTO 2015*, pages 585–605, Santa Barbara, CA, USA, 2015. Springer Berlin Heidelberg.

[130] M. S. Eddine, M. A. Ferrag, O. Friha, and L. Maglaras. Easbf: An efficient authentication scheme over blockchain for fog computing-enabled internet of vehicles. *Journal of Information Security and Applications*, 59:102802, 2021.

[131] O. Elijah, T. A. Rahman, I. Orikumhi, C. Y. Leow, and M. N. Hindia. An Overview of Internet of Things (IoT) and Data Analytics in Agriculture: Benefits and Challenges. *IEEE Internet of Things Journal*, 5(5):3758–3773, 2018.

[132] S. Escolar, F. Rincón, X. del Toro, J. Barba, F. J. Villanueva, M. J. Santofimia, D. Villa, and J. C. López. The PLATINO Experience: A LoRa-based Network of Energy-Harvesting Devices for Smart Farming. In *XXXIV Conference on Design of Circuits and Integrated Systems (DCIS)*, pages 1–6, 2019.

[133] Q. Fan, J. Chen, L. J. Deborah, and M. Luo. A secure and efficient authentication and data sharing scheme for internet of things based on blockchain. *Journal of Systems Architecture*, 117:102112, 2021.

[134] M. S. Farooq, S. Riaz, A. Abid, K. Abid, and M. A. Naeem. A Survey on the Role of IoT in Agriculture for the Implementation of Smart Farming. *IEEE Access*, 7:156237–156271, 2019.

[135] H. Feng, J. Wang, and Y. Li. An Efficient Blockchain Transaction Retrieval System. *Future Internet*, 14(9), 2022.

[136] H. Feng, X. Wang, Y. Duan, J. Zhang, and X. Zhang. Applying blockchain technology to improve agri-food traceability: A review of development methods, benefits and challenges. *Journal of Cleaner Production*, 260:121031, 2020.

[137] Q. Feng, D. He, S. Zeadally, M. K. Khan, and N. Kumar. A Survey on Privacy Protection in Blockchain System. *Journal of Network and Computer Applications*, 126:45 – 58, 2019.

[138] M. A. Ferrag, L. Shu, H. Djallel, and K.-K. R. Choo. Deep Learning-Based Intrusion Detection for Distributed Denial of Service Attack in Agriculture 4.0. *Electronics*, 10(11), 2021.

[139] M. A. Ferrag, L. Shu, X. Yang, A. Derhab, and L. Maglaras. Security and Privacy for Green IoT-Based Agriculture: Review, Blockchain Solutions, and Challenges. *IEEE Access*, 8:32031–32053, 2020.

[140] B. A. Forouzan and D. Mukhopadhyay. *Cryptography and Network Security*, volume 12. Mc Graw Hill Education (India) Private Limited New York, NY, USA:, 2015.

[141] O. Friha, M. A. Ferrag, L. Shu, and M. Nafa. A Robust Security Framework based on Blockchain and SDN for Fog Computing enabled Agricultural Internet of Things. In *2020 International Conference on Internet of Things and Intelligent Applications (ITIA)*, pages 1–5, 2020.

[142] C. M. Galanakis. The Food Systems in the Era of the Coronavirus (COVID-19) Pandemic Crisis. *Foods*, 9(4):523, 2020.

[143] F. Gandino, B. Montrucchio, M. Rebaudengo, and E. R. Sanchez. On Improving Automation by Integrating RFID in the Traceability Management of the Agri-Food Sector. *IEEE Transactions on Industrial Electronics*, 56(7):2357–2365, 2009.

[144] L. Ge, C. Brewster, J. Spek, A. Smeenk, J. Top, F. van Diepen, B. Klaase, C. Graumans, and M. d. R. de Wildt. *Blockchain for agriculture and food: Findings from the pilot study.* Number 2017-112. Wageningen Economic Research, 2017.

[145] R. Gebbers and V. I. Adamchuk. Precision agriculture and food security. *Science*, 327(5967):828–831, 2010.

[146] B. Gernert, S. Rottmann, and L. C. Wolf. PotatoMesh: A Solar Powered WSN Testbed: Poster. In *17th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, MobiHoc '16, pages 391–392, Paderborn, Germany, 2016.

[147] S. Ghosh and A. Roy. Desert Locust in India: The 2020 invasion and associated risks. 2020.

[148] H. Gilbert and H. Handschuh. Security Analysis of SHA-256 and Sisters. In *International Workshop on Selected Areas in Cryptography (SAC'03)*, pages 175–193, Ottawa, Canada, 2004.

[149] D. Glaroudis, A. Iossifides, and P. Chatzimisios. Survey, comparison and research challenges of IoT application protocols for smart farming. *Computer Networks*, 168:107037, 2020.

[150] P. Gonzalez-De-Santos, R. Fernández, D. Sepúlveda, E. Navas, and M. Armada. Unmanned ground vehicles for smart farms. *Agronomy-Climate Change & Food Security*, page 73, 2020.

[151] P. Gope, A. K. Das, N. Kumar, and Y. Cheng. Lightweight and Physically Secure Anonymous Mutual Authentication Protocol for Real-Time Data Access in Industrial Wireless Sensor Networks. *IEEE Transactions on Industrial Informatics*, 15(9):4957–4968, 2019.

[152] G. B. Gregorioa and R. C. Ancog. Assessing the Impact of the COVID-19 Pandemic on Agricultural Production in Southeast Asia: Toward Transformative Change in Agricultural Food Systems. *Asian Journal of Agriculture and Development*, (1362-2020-1097), Jun 2020.

[153] D. Gunasekera and E. Valenzuela. Adoption of Blockchain Technology in the Australian Grains Trade: An Assessment of Potential Economic Effects. *Economic Papers: A journal of applied economics and policy*.

[154] D. S. Gupta, S. H. Islam, M. S. Obaidat, P. Vijayakumar, N. Kumar, and Y. Park. A Provably Secure and Lightweight Identity-Based Two-Party Authenticated Key Agreement Protocol for IIoT Environments. *IEEE Systems Journal*, 15(2):1732–1741, 2021.

[155] M. Gupta, M. Abdelsalam, S. Khorsandroo, and S. Mittal. Security and Privacy in Smart Farming: Challenges and Opportunities. *IEEE Access*, 8:34564–34584, 2020.

[156] R. Gupta, A. Kumari, and S. Tanwar. A taxonomy of blockchain envisioned edge-as-a-connected autonomous vehicles. *Transactions on Emerging Telecommunications Technologies*, page e4009, 2020.

[157] R. Gupta, S. Tanwar, F. Al-Turjman, P. Italiya, A. Nauman, and S. W. Kim. Smart Contract Privacy Protection Using AI in Cyber-Physical Systems: Tools, Techniques and Challenges. *IEEE Access*, 8:24746–24772, 2020.

[158] R. Gupta, S. Tanwar, S. Tyagi, and N. Kumar. Tactile internet and its applications in 5G era: A comprehensive review. *International Journal of Communication Systems*, 32(14):e3981, 2019.

[159] R. Gupta, S. Tanwar, S. Tyagi, and N. Kumar. Machine Learning Models for Secure Data Analytics: A taxonomy and threat model. *Computer Communications*, 153:406 – 440, 2020.

[160] R. Gupta, S. Tanwar, S. Tyagi, N. Kumar, M. S. Obaidat, and B. Sadoun. HaBiTs: Blockchain-based Telesurgery Framework for Healthcare 4.0. In *International Conference on Computer, Information and Telecommunication Systems (CITS)*, pages 1–5, Beijing, China, China, 2019.

[161] J. Harris, L. Depenbusch, A. A. Pal, R. M. Nair, and S. Ramasamy. Food system disruption: initial livelihood and dietary effects of COVID-19 on vegetable producers in India. *Food Security*, 12(4):841–851, 2020.

[162] R. Hartung, U. Kulau, B. Gernert, S. Rottmann, and L. Wolf. On the Experiences with Testbeds and Applications in Precision Farming. In *First ACM International Workshop on the Engineering of Reliable, Robust, and Secure Embedded Wireless Sensing Systems*, pages 54–61, Delft, Netherlands, 2017.

[163] V. Hassija, S. Batra, V. Chamola, T. Anand, P. Goyal, N. Goyal, and M. Guizani. A blockchain and deep neural networks-based secure framework for enhanced crop protection. *Ad Hoc Networks*, 119:102537, 2021.

[164] V. Hassija, V. Chamola, V. Gupta, S. Jain, and N. Guizani. A Survey on Supply Chain Security: Application Areas, Security Threats, and Solution Architectures. *IEEE Internet of Things Journal*, 8(8):6222–6246, 2021.

[165] J. Hathaliya, P. Sharma, S. Tanwar, and R. Gupta. Blockchain-Based Remote Patient Monitoring in Healthcare 4.0. In *9th IEEE International Conference on Advanced Computing (IACC)*, pages 87–91, Tiruchirappalli, India, 2019.

[166] D. He, S. Zeadally, B. Xu, and X. Huang. An Efficient Identity-Based Conditional Privacy-Preserving Authentication Scheme for Vehicular Ad Hoc Networks. *IEEE Transactions on Information Forensics and Security*, 10:2681–2691, 2015.

[167] C. Hedley. The role of precision agriculture for improved nutrient management on farms. *Journal of the Science of Food and Agriculture*, 95(1):12–19, 2015.

[168] J. Hu, M. J. Reed, M. Al-Naday, and N. Thomos. Hybrid Blockchain for IoT–Energy Analysis and Reward Plan. *Sensors*, 21(1):1–21, 2021.

[169] Y. . Hu, A. Perrig, and D. B. Johnson. Packet leashes: a defense against wormhole attacks in wireless networks. In *IEEE INFOCOM 2003. Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 3, pages 1976–1986, San Francisco, California, USA, 2003.

[170] S. L. Inc. Storj: A Decentralized Cloud Storage Network Framework - WhitePaper. https://storj.io/storj.pdf.

[171] L. Insights. India's Coffee Board registers 30,000 farmers on blockchain marketplace. https://www.ledgerinsights.com/india-coffee-board-blockchain-marketplace/.

[172] Intel Corporation. Hyperledger Sawtooth Architecture Guide, 2022. https://sawtooth.hyperledger.org/docs/core/releases/1.1/architecture.html.

[173] A. Islam and S. Y. Shin. A blockchain-based secure healthcare scheme with the assistance of unmanned aerial vehicle in internet of things. *Computers & Electrical Engineering*, 84:106627, 2020.

[174] S. Itoo, A. A. Khan, V. Kumar, A. Alkhayyat, M. Ahmad, and J. Srinivas. CKMIB: Construction of Key Agreement Protocol for Cloud Medical Infrastructure Using Blockchain. *IEEE Access*, 10:67787–67801, 2022.

[175] M. M. Jahn, W. L. Oemichen, G. F. Treverton, S. L. David, M. A. Rose, M. A. Brosig, B. J. Jayamah, W. K. Hutchison, and B. B.Rimestad. Cyber Risk and Security Implications in Smart Agriculture and Food Systems, 2019. https://jahnresearchgroup.webhosting.cals.wisc.edu/wp-content/uploads/sites/223/2019/01/Agricultural-Cyber-Risk-and-Security.pdf.

[176] A. K. Jain, A. Ross, and S. Prabhakar. An Introduction to Biometric Recognition. *IEEE Transactions on circuits and systems for video technology*, 14(1):4–20, 2004.

[177] D. S. Jat, A. S. Limbo, and C. Singh. Internet of things for automation in smart agriculture: a technical review. In *Smart Farming Technologies for Sustainable Agricultural Development*, pages 93–105. IGI Global, 2019.

[178] H. M. Jawad, R. Nordin, S. K. Gharghan, A. M. Jawad, and M. Ismail. Energy-Efficient Wireless Sensor Networks for Precision Agriculture: A Review. *Sensors*, 17(8), 2017.

[179] X. Jia, M. Luo, H. Wang, J. Shen, and D. He. A Blockchain-Assisted Privacy-Aware Authentication Scheme for Internet of Medical Things. *IEEE Internet of Things Journal*, 2022.

[180] J. Jiang and M. Moallem. Development of an Intelligent LED Lighting Control Testbed for IoT-based Smart Greenhouses. In *IECON 2020 The 46th Annual Conference of the IEEE Industrial Electronics Society*, pages 5226–5231, 2020.

[181] Q. Jiang, S. Zeadally, J. Ma, and D. He. Lightweight three-factor authentication and key agreement protocol for internet-integrated wireless sensor networks. *IEEE Access*, 5:3376–3392, 2017.

[182] W. Jiang, H. Li, G. Xu, M. Wen, G. Dong, and X. Lin. A Privacy-Preserving Thin-Client Scheme in Blockchain-Based PKI. In *37th IEEE Global Communications Conference (GLOBECOM)*, pages 1–6, Abu Dhabi, United Arab Emirates, 2018.

[183] W. Jiang, H. Li, G. Xu, M. Wen, G. Dong, and X. Lin. PTAS: Privacy-preserving Thin-client Authentication Scheme in blockchain-based PKI. *Future Gener. Comput. Syst.*, 96:185–195, 2019.

[184] A. T. B. Jin, D. N. C. Ling, and A. Goh. Biohashing: two factor authentication featuring fingerprint data and tokenised random number. *Pattern Recognition*, 37(11):2245–2255, 2004.

[185] A. Jindal, G. S. Aujla, and N. Kumar. SURVIVOR: A blockchain based edge-as-a-service framework for secure energy trading in SDN-enabled vehicle-to-grid environment. *Computer Networks*, 153:36 – 48, 2019.

[186] D. Johnson, A. Menezes, and S. Vanstone. The Elliptic Curve Digital Signature Algorithm (ECDSA). *International Journal of Information Security*, 1(1):36–63, 2001.

[187] G. Karame. On the Security and Scalability of Bitcoin's Blockchain. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*, pages 1861–1862, 2016.

[188] N. Karthickraja, V. Sumathy, and M. Jabeer Ahamed. A novel hybrid routing protocol for data aggregation in agricultural applications. In *International Conference on Communication Control and Computing Technologies*, pages 227–231, 2010.

[189] K. Kaur, S. Garg, G. Kaddoum, F. Gagnon, and S. H. Ahmed. Blockchain-Based Lightweight Authentication Mechanism for Vehicular Fog Infrastructure. In *IEEE International Conference on Communications Workshops (ICC Workshops)*, pages 1–6, Shanghai, China, 2019.

[190] A. Khanna and S. Kaur. Evolution of Internet of Things (IoT) and its significant impact in the field of Precision Agriculture. *Computers and Electronics in Agriculture*, 157:218 – 231, 2019.

[191] K. Khullar. Implementing PBFT in Blockchain, 2019. https://medium.com/coinmonks/implementing-pbft-in-blockchain-12368c6c9548.

[192] A. Kiayias, A. Russell, B. David, and R. Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. In *Advances in Cryptology (CRYPTO'17)*, pages 357–388, Santa Barbara, CA, USA, 2017.

[193] S. Kim, Y. Kwon, and S. Cho. A Survey of Scalability Solutions on Blockchain. In *International Conference on Information and Communication Technology Convergence (ICTC)*, pages 1204–1207, 2018.

[194] Y. Kim, P. Bae, J. Han, and Y.-B. Ko. Data aggregation in precision agriculture for low-power and lossy networks. In *IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM)*, pages 438–443, 2015.

[195] S. King and S. Nadal. PPCoin: Peer-to-Peer Crypto-Currency with Proof-of-Stake, 2012.

[196] L. Klerkx, E. Jakku, and P. Labarthe. A review of social science on digital agriculture, smart farming and agriculture 4.0: New contributions and a future research agenda. *NJAS - Wageningen Journal of Life Sciences*, 90-91:100315, 2019.

[197] D. E. Knuth. *The Art of Computer Programming: Seminumerical Algorithms*, volume 2. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 3rd edition, 1997.

[198] N. Koblitz, A. Menezes, and S. Vanstone. The state of elliptic curve cryptography. *Designs, codes and cryptography*, 19(2-3):173–193, 2000.

[199] P. C. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In *16th Annual International Cryptology Conference – Advances in Cryptology (CRYPTO'96)*, pages 104–113, Barbara, California, USA, 1996.

[200] P. C. Kocher, J. Jaffe, and B. Jun. Differential Power Analysis. In *19th Annual International Cryptology Conference on Advances in Cryptology (CRYPTO'99)*, pages 388–397, Santa Barbara, California, USA, 1999.

[201] A. Kong, K.-H. Cheung, D. Zhang, M. Kamel, and J. You. An analysis of BioHashing and its variants. *Pattern Recognition*, 39(7):1359–1368, 2006.

[202] A. Kosba, A. Miller, E. Shi, Z. Wen, and C. Papamanthou. Hawk: The Blockchain Model of Cryptography and Privacy-Preserving Smart Contracts. In *37th IEEE Symposium on Security and Privacy (S&P'16)*, pages 839–858, Fairmont,San Jose, CA, 2016.

[203] P. Koshy, D. Koshy, and P. McDaniel. An Analysis of Anonymity in Bitcoin Using P2P Network Traffic. In *18th International Conference on Financial Cryptography and Data Security*, pages 469–485, Christ Church, Barbados, 2014.

[204] U. Kulau, S. Schildt, S. Rottmann, B. Gernert, and L. Wolf. Demo: PotatoNet – Robust Outdoor Testbed for WSNs: Experiment like on Your Desk. Outside. pages 59–60, Paris, France, 2015.

[205] A. Kumari, R. Gupta, S. Tanwar, and N. Kumar. Blockchain and AI amalgamation for energy cloud management: Challenges, solutions, and future directions. *Journal of Parallel and Distributed Computing*, 143:148 – 166, 2020.

[206] A. Kumari, S. Tanwar, S. Tyagi, and N. Kumar. Fog computing for Healthcare 4.0 environment: Opportunities and challenges. *Computers & Electrical Engineering*, 72:1 – 13, 2018.

[207] A. Kumari, S. Tanwar, S. Tyagi, and N. Kumar. Verification and validation techniques for streaming big data analytics in Internet of Things environment. *IET Networks*, 8(3):155–163, 2019.

[208] A. Kumari, S. Tanwar, S. Tyagi, N. Kumar, M. S. Obaidat, and J. J. P. C. Rodrigues. Fog Computing for Smart Grid Systems in the 5G Environment: Challenges and Solutions. *IEEE Wireless Communications*, 26(3):47–53, 2019.

[209] A. Kumari, S. Tanwar, S. Tyagi, N. Kumar, R. M. Parizi, and K.-K. R. Choo. Fog data analytics: A taxonomy and process model. *Journal of Network and Computer Applications*, 128:90 – 104, 2019.

[210] J. Kwon. Tendermint: Consensus without mining. *Self-Published Paper (Draft v.0.6)*, 1(11), 2014. https://tendermint.com/static/docs/tendermint.pdf.

[211] L. Lamport, R. Shostak, and M. Pease. The Byzantine Generals Problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, 1982.

[212] M. Lezoche, J. E. Hernandez, M. del Mar Eva Alemany Díaz, H. Panetto, and J. Kacprzyk. Agri-food 4.0: A survey of the supply chains and technologies for the future agriculture. *Computers in Industry*, 117:103187, 2020.

[213] S. Li, L. Da Xu, and S. Zhao. The Internet of Things: A Survey. *Information Systems Frontiers*, 17(2):243–259, 2015.

[214] X. Li, S. Liu, F. Wu, S. Kumari, and J. J. P. C. Rodrigues. Privacy Preserving Data Aggregation Scheme for Mobile Edge Computing Assisted IoT Applications. *IEEE Internet of Things Journal*, 6(3):4755–4763, 2019.

[215] X. Li, T. Liu, M. S. Obaidat, F. Wu, P. Vijayakumar, and N. Kumar. A Lightweight Privacy-Preserving Authentication Protocol for VANETs. *IEEE Systems Journal*, pages 1–11, 2020.

[216] X. Li, J. Niu, S. Kumari, F. Wu, and K.-K. R. Choo. A robust biometrics based three-factor authentication scheme for Global Mobility Networks in smart city. *Future Generation Computer Systems*, 83:607–618, 2018.

[217] D.-Y. Lin, C.-J. Juan, and C.-C. Chang. Managing Food Safety With Pricing, Contracts and Coordination in Supply Chains. *IEEE Access*, 7:150892–150909, 2019.

[218] Y. Lin, X. Wang, Q. Gan, and M. Yao. A secure cross-domain authentication scheme with perfect forward security and complete anonymity in fog computing. *Journal of Information Security and Applications*, 63:103022, 2021.

[219] Y.-P. Lin, J. R. Petway, J. Anthony, H. Mukhtar, S.-W. Liao, C.-F. Chou, and Y.-F. Ho. Blockchain: The Evolutionary Next Step for ICT E-Agriculture. *Environments*, 4(3), 2017.

[220] J. Liu, R. Liu, and Y. Lai. Risk-Based Dynamic Identity Authentication Method Based on the UCON Model. *Security and Communication Networks*, 2022, 2022.

[221] Y. Liu, X. Ma, L. Shu, G. P. Hancke, and A. M. Abu-Mahfouz. From Industry 4.0 to Agriculture 4.0: Current Status, Enabling Technologies, and Research Challenges. *IEEE Transactions on Industrial Informatics*, 17(6):4322–4334, 2021.

[222] C. Lu. Russia's Invasion Unleashes 'Perfect Storm' in Global Agriculture, 2022.

[223] A. Lumini and L. Nanni. An improved BioHashing for human authentication. *Pattern Recognition*, 40(3):1057–1065, 2007.

[224] W. H. Maes and K. Steppe. Perspectives for Remote Sensing with Unmanned Aerial Vehicles in Precision Agriculture. *Trends in Plant Science*, 24(2):152–164, 2019.

[225] G. Malavolta, P. Moreno-Sanchez, C. Schneidewind, A. Kate, and M. Maffei. Anonymous Multi-Hop Locks for Blockchain Scalability and Interoperability. In *Network and Distributed Systems Security (NDSS) Symposium*, 2019.

[226] M. Mammarella, L. Comba, A. Biglia, F. Dabbene, and P. Gay. Cooperative Agricultural Operations of Aerial and Ground Unmanned Vehicles. In *IEEE International Workshop on Metrology for Agriculture and Forestry (MetroAgriFor)*, pages 224–229, 2020.

[227] S. Mandal, B. Bera, A. K. Sutrala, A. K. Das, K. R. Choo, and Y. Park. Certificateless-signcryption-based three-factor user access control scheme for iot environment. *IEEE Internet of Things Journal*, 7(4):3184–3197, 2020.

[228] I. Marcu, C. Voicu, A. M. C. Drăgulinescu, O. Fratu, G. Suciu, C. Balaceanu, and M. M. Andronache. Overview of IoT Basic Platforms for Precision Agriculture. In *Future*

Access Enablers for Ubiquitous and Intelligent Infrastructures, pages 124–137. Springer International Publishing, 2019.

[229] R. Martínez, J. A. Pastor, B. Álvarez, and A. Iborra. A Testbed to Evaluate the FIWARE-Based IoT Platform in the Domain of Precision Agriculture. *Sensors*, 16(11), 2016.

[230] R. Martino and A. Cilardo. Designing a SHA-256 processor for blockchain-based IoT applications. *Internet of Things*, 11:100254, 2020.

[231] I. D. Mastan and S. Paul. A New Approach to Deanonymization of Unreachable Bitcoin Nodes. In *17th International Conference on Cryptology and Network Security*, pages 277–298, Naples, Italy, 2018.

[232] W. E. May. Secure Hash Standard.FIPS PUB 180-1, National Institute of Standards and Technology (NIST), U.S. Department of Commerce, April 1995. Accessed on January 2020, 2015. http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf.

[233] P. Mehta, R. Gupta, and S. Tanwar. Blockchain envisioned UAV networks: Challenges, solutions, and comparisons. *Computer Communications*, 151:518 – 538, 2020.

[234] S. Meiklejohn, M. Pomarole, G. Jordan, K. Levchenko, D. McCoy, G. M. Voelker, and S. Savage. A Fistful of Bitcoins: Characterizing Payments among Men with No Names. In *ACM SIGCOMM Conference on Internet Measurement Conference*, pages 127–140, 2013.

[235] T. S. Messerges, E. A. Dabbish, and R. H. Sloan. Examining smart-card security under the threat of power analysis attacks. *IEEE Transactions on Computers*, 51(5):541–552, 2002.

[236] L. Miloudi, K. Rezeg, O. Kazar, and M. K. Miloudi. Smart Sustainable Farming Management Using Integrated Approach of IoT, Blockchain & Geospatial Technologies. In M. Ezziyyani, editor, *Advanced Intelligent Systems for Sustainable Development (AI2SD'2019)*, pages 340–347, Marrakech, Morocco, 2020. Springer International Publishing.

[237] M. Milutinovic, W. He, H. Wu, and M. Kanwal. Proof of Luck: An Efficient Blockchain Consensus Protocol. In *1st Workshop on System Software for Trusted Execution (SysTEX'16)*, number 2, pages 1–6, Tento, Italy, 2016.

[238] D. Miorandi, S. Sicari, F. De Pellegrini, and I. Chlamtac. Internet of things: Vision, applications and research challenges. *Ad Hoc Networks*, 10(7):1497 – 1516, 2012.

[239] D. Mishra, A. K. Das, and S. Mukhopadhyay. A secure and efficient ECC-based user anonymity-preserving session initiation authentication protocol using smart card. *Peer-to-Peer Networking and Applications*, 9(1):171–192, 2016.

[240] D. Mishra, D. Dharminder, P. Yadav, Y. S. Rao, P. Vijayakumar, and N. Kumar. A provably secure dynamic ID-based authenticated key agreement framework for mobile edge computing without a trusted party. *Journal of Information Security and Applications*, 55:102648, 2020.

[241] I. Mistry, S. Tanwar, S. Tyagi, and N. Kumar. Blockchain for 5G-enabled IoT for industrial automation: A systematic review, solutions, and challenges. *Mechanical Systems and Signal Processing*, 135:106382, 2020.

[242] A. Mitra, B. Bera, A. K. Das, S. S. Jamal, and I. You. Impact on blockchain-based AI/ML-enabled big data analytics for Cognitive Internet of Things environment. *Computer Communications*, 197:173–185, 2023.

[243] U. R. Mogili and B. B. V. L. Deepak. Review on Application of Drone Systems in Precision Agriculture. *Procedia Computer Science*, 133:502–509, 2018.

[244] D. J. Mulla. Twenty five years of remote sensing in precision agriculture: Key advances and remaining knowledge gaps. *Biosystems Engineering*, 114(4):358–371, 2013.

[245] S. Nakamoto. Bitcoin open source implementation of p2p currency. *P2P foundation*, 18, 2009.

[246] A. Narayanan, J. Bonneau, E. Felten, A. Miller, and S. Goldfeder. *Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction*. Princeton University Press, USA, 2016.

[247] J. Newsome, E. Shi, D. Song, and A. Perrig. The Sybil attack in sensor networks: analysis & defenses. In *Third International Symposium on Information Processing in Sensor Networks, 2004. IPSN 2004*, pages 259–268, Berkeley, California, USA, 2004.

[248] G.-T. Nguyen and K. Kim. A Survey about Consensus Algorithms Used in Blockchain. *Journal of Information Processing Systems*, 14:101–128, 2018.

[249] V. Odelu, A. K. Das, and A. Goswami. A Secure Biometrics-Based Multi-Server Authentication Protocol Using Smart Cards. *IEEE Transactions on Information Forensics and Security*, 10(9):1953–1966, 2015.

[250] V. Odelu, A. K. Das, and A. Goswami. SEAP: Secure and efficient authentication protocol for NFC applications using pseudonyms. *IEEE Transactions on Consumer Electronics*, 62(1):30–38, 2016.

[251] M. of Electronics and G. o. I. Information Technology. Centre of Excellence in Blockchain Technology, National Informatics Centre. https://blockchain.gov.in/.

[252] M. of Skill Development and Entrepreneurship. IoT Technician for Smart Agriculture.

[253] S. S. Panda, D. Jena, B. K. Mohanta, S. Ramasubbareddy, M. Daneshmand, and A. H. Gandomi. Authentication and Key Management in Distributed IoT Using Blockchain Technology. *IEEE Internet of Things Journal*, 8(16):12947–12954, 2021.

[254] B. Parno, A. Perrig, and V. Gligor. Distributed detection of node replication attacks in sensor networks. In *26th IEEE Symposium on Security and Privacy (S&P'05)*, pages 49–63, Oakland, California, USA, 2005.

[255] F. J. Pierce and P. Nowak. Aspects of Precision Agriculture. volume 67 of *Advances in Agronomy*, pages 1–85. Academic Press, 1999.

[256] D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, 2000.

[257] N. Potlapally, S. Ravi, A. Raghunathan, and N. Jha. A study of the energy consumption characteristics of cryptographic algorithms and security protocols. *IEEE Transactions on Mobile Computing*, 5(2):128–143, 2006.

[258] N. R. Potlapally, S. Ravi, A. Raghunathan, and N. K. Jha. Analyzing the energy consumption of security protocols. In *Proceedings of the 2003 international symposium on Low power electronics and design*, pages 30–35, 2003.

[259] S. Prabhakar, S. Pankanti, and A. K. Jain. Biometric recognition: Security and Privacy Concerns. *IEEE security & privacy*, 1(2):33–42, 2003.

[260] T. H. Pranto, A. A. Noman, A. Mahmud, and A. B. Haque. Blockchain and smart contract for iot enabled smart agriculture. *PeerJ Computer Science*, 7:e407, 2021.

[261] M. Pu and Y. Zhong. Rising concerns over agricultural production as COVID-19 spreads: Lessons from China. *Global Food Security*, 26:100409, 2020.

[262] D. Pujara, P. Kukreja, and S. Gajjar. Design and Development of E-Sense: IoT based Environment Monitoring System. In *IEEE Students Conference on Engineering Systems (SCES)*, pages 1–5, 2020.

[263] L. Raja and S. Vyas. The study of technological development in the field of smart farming. In *Smart Farming Technologies for Sustainable Agricultural Development*, pages 1–24. IGI Global, 2019.

[264] R. Ramakumar. Agriculture and the Covid-19 Pandemic: An Analysis with special reference to India. *Review of Agrarian Studies*, 10(2369-2020-1856), 2020.

[265] D. Rangwani, D. Sadhukhan, S. Ray, M. K. Khan, and M. Dasgupta. An improved privacy preserving remote user authentication scheme for agricultural wireless sensor network. *Transactions on Emerging Telecommunications Technologies*, 32(3):e4218, 2021.

[266] P. P. Ray. Internet of things for smart agriculture: Technologies, practices and future direction. *Journal of Ambient Intelligence and Smart Environments*, 9(4):395–420, 2017.

[267] A. Rettore de Araujo Zanella, E. da Silva, and L. C. Pessoa Albini. Security challenges to smart agriculture: Current state, key issues, and future directions. *Array*, 8:100048, 2020.

[268] A. B. Review. Tea Board of India to Adopt Blockchain Technology for Tea Traceability. https://www.asiablockchainreview.com/tea-board-of-india-to-adopt-blockchain-technology-for-tea-traceability/.

[269] L. P. Reynolds, M. C. Wulster-Radcliffe, D. K. Aaron, and T. A. Davis. Importance of Animals in Agricultural Sustainability and Food Security. *The Journal of Nutrition*, 145(7):1377–1379, 2015.

[270] S. Roy, S. Chatterjee, A. K. Das, S. Chattopadhyay, N. Kumar, and A. V. Vasilakos. On the Design of Provably Secure Lightweight Remote User Authentication Scheme for Mobile Cloud Computing Services. *IEEE Access*, 5:25808–25825, 2017.

[271] A. Roychoudhury. Desert Locust: A Menace to Indian Agriculture and Economy. *Young Scientist-Tomorrow's Science Begins Today*, 4(1):14–20, 2020.

[272] J. Ruan, Y. Wang, F. T. S. Chan, X. Hu, M. Zhao, F. Zhu, B. Shi, Y. Shi, and F. Lin. A Life Cycle Framework of Green IoT-Based Agriculture and Its Finance, Operation, and Management Issues. *IEEE Communications Magazine*, 57(3):90–96, 2019.

[273] L. Ruiz-Garcia and L. Lunadei. The role of RFID in agriculture: Applications, limitations and challenges. *Computers and Electronics in Agriculture*, 79(1):42–50, 2011.

[274] D. Sadhukhan, S. Ray, G. Biswas, M. Khan, and M. Dasgupta. A lightweight remote user authentication scheme for IoT communication using elliptic curve cryptography. *Journal of Supercomputing*, 2020.

[275] S. Sadowski and P. Spachos. Solar-Powered Smart Agricultural Monitoring System Using Internet of Things Devices. In *IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, pages 18–23, 2018.

[276] V. Saiz-Rubio and F. Rovira-Más. From smart farming towards agriculture 5.0: a review on crop data management. *Agronomy*, 10(2):207, 2020.

[277] K. Salah, N. Nizamuddin, R. Jayaraman, and M. Omar. Blockchain-Based Soybean Traceability in Agricultural Supply Chain. *IEEE Access*, 7:73295–73305, 2019.

[278] V. Salin. Information technology in agri-food supply chains. *The International Food and Agribusiness Management Review*, 1(3):329–334, 1998.

[279] S. Sankar, P. Srinivasan, A. K. Luhach, R. Somula, and N. Chilamkurti. Energy-aware grid-based data aggregation scheme in routing protocol for agricultural internet of things. *Sustainable Computing: Informatics and Systems*, 28:100422, 2020.

[280] A. Sapirshtein, Y. Sompolinsky, and A. Zohar. Optimal Selfish Mining Strategies in Bitcoin. In *20th International Conference on Financial Cryptography and Data Security*, pages 515–532. Springer Berlin Heidelberg, 2017.

[281] D. Schwartz, N. Youngs, A. Britto, et al. The ripple protocol consensus algorithm. *Ripple Labs Inc White Paper*, 5(8), 2014.

[282] U. Shafi, R. Mumtaz, J. Garcia-Nieto, S. A. Hassan, S. A. R. Zaidi, and N. Iqbal. Precision Agriculture Techniques and Practices: From Considerations to Applications. *Sensors*, 19(17), 2019.

[283] P. K. Sharma, N. Kumar, and J. H. Park. Blockchain-Based Distributed Framework for Automotive Industry in a Smart City. *IEEE Transactions on Industrial Informatics*, 15(7):4197–4205, 2019.

[284] Q. ShenTu and J. Yu. Research on anonymization and de-anonymization in the bitcoin system. *Computing Research Repository (CoRR)*, abs/1510.07782, 2015.

[285] X. Shi, X. An, Q. Zhao, H. Liu, L. Xia, X. Sun, and Y. Guo. State-of-the-Art Internet of Things in Protected Agriculture. *Sensors*, 19(8), 2019.

[286] M. Shuai, L. Xiong, C. Wang, and N. Yu. A secure authentication scheme with forward secrecy for industrial internet of things using Rabin cryptosystem. *Computer Communications*, 160:215 – 227, 2020.

[287] J. H. Silverman. *The arithmetic of elliptic curves*, volume 106. Springer, 2009.

[288] R. Singh, S. Tanwar, and T. P. Sharma. Utilization of blockchain for mitigating the distributed denial of service attacks. *Security and Privacy*, 3(3):e96, 2020.

[289] S. K. Singh, S. Rathore, and J. H. Park. BlockIoTIntelligence: A Blockchain-enabled Intelligent IoT Architecture with Artificial Intelligence. *Future Generation Computer Systems*, 110:721 – 743, 2020.

[290] R. S. Sinha, Y. Wei, and S.-H. Hwang. A survey on LPWA technology: LoRa and NB-IoT. *ICT Express*, 3(1):14–21, 2017.

[291] C. Smith. B Merkle Tree. https://medium.com/proxima-one/b-merkle-tree-824952837bfe.

[292] C. Smith and A. Rusnak. Dynamic Merkle B-tree with Efficient Proofs, 2020.

[293] J. Song, Q. Zhong, W. Wang, C. Su, Z. Tan, and Y. Liu. FPDP:Flexible Privacy-preserving Data Publishing Scheme for Smart Agriculture. *IEEE Sensors Journal*, 2020.

[294] S. Sontowski, M. Gupta, S. S. Laya Chukkapalli, M. Abdelsalam, S. Mittal, A. Joshi, and R. Sandhu. Cyber Attacks on Smart Farming Infrastructure. In *IEEE 6th International Conference on Collaboration and Internet Computing (CIC)*, pages 135–143, 2020.

[295] J. Srinivas, A. K. Das, N. Kumar, and J. Rodrigues. Cloud Centric Authentication for Wearable Healthcare Monitoring System. *IEEE Transactions on Dependable and Secure Computing*, 2018.

[296] J. Srinivas, A. K. Das, N. Kumar, and J. J. P. C. Rodrigues. TCALAS: Temporal Credential-Based Anonymous Lightweight Authentication Scheme for Internet of Drones Environment. *IEEE Transactions on Vehicular Technology*, 68(7):6903–6916, 2019.

[297] A. Srinivasan. *Handbook of Precision Agriculture: Principles and Applications*. CRC press, 2006.

[298] J. V. Stafford. Implementing Precision Agriculture in the 21st Century. *Journal of Agricultural Engineering Research*, 76(3):267–275, 2000.

[299] W. Stallings. *Cryptography and Network Security: Principles and Practices*. Pearson Education, India, 6th edition, 2014.

[300] G. Stamatescu, C. Dragana, I. Stamatescu, L. Ichim, and D. Popescu. IoT-Enabled Distributed Data Processing for Precision Agriculture. In *27th Mediterranean Conference on Control and Automation (MED)*, pages 286–291, 2019.

[301] I. Stewart. Proof of Burn - Bitcoin Wiki, 2012. https://en.bitcoin.it/wiki/Proof_of_burn.

[302] D. R. Stinson. *Cryptography: Theory and Practice*. Chapman and Hall/CRC, 2005.

[303] M. Swain, D. Zimon, R. Singh, M. F. Hashmi, M. Rashid, and S. Hakak. LoRa-LBO: An Experimental Analysis of LoRa Link Budget Optimization in Custom Build IoT Test Bed for Agriculture 4.0. *Agronomy*, 11(5), 2021.

[304] Szabo, Nick. Smart Contracts. http://szabo.best.vwh.net/smart.contracts.html.

[305] W. L. Tai, Y. F. Chang, and W. H. Li. An IoT notion-based authentication and key agreement scheme ensuring user anonymity for heterogeneous ad hoc wireless sensor networks. *Journal of Information Security and Applications*, 34:133 – 141, 2017.

[306] J. M. Talavera, L. E. Tobon, J. A. Gomez, M. A. Culman, J. M. Aranda, D. T. Parra, L. A. Quiroz, A. Hoyos, and L. E. Garreta. Review of IoT applications in agro-industrial and environmental fields. *Computers and Electronics in Agriculture*, 142:283 – 297, 2017.

[307] S. Tanwar, Q. Bhatia, P. Patel, A. Kumari, P. K. Singh, and W. Hong. Machine Learning Adoption in Blockchain-Based Smart Applications: The Challenges, and a Way Forward. *IEEE Access*, 8:474–488, 2020.

[308] S. Tanwar, K. Parekh, and R. Evans. Blockchain-based electronic healthcare record system for healthcare 4.0 applications. *Journal of Information Security and Applications*, 50:102407, 2020.

[309] S. Tanwar, S. Tyagi, and S. Kumar. The Role of Internet of Things and Smart Grid for the Development of a Smart City. In *Intelligent Communication and Computational Technologies*, pages 23–33, Gandhinagar, Gujarat, India, 2018.

[310] S. Tanwar, J. Vora, S. Tyagi, N. Kumar, and M. S. Obaidat. A systematic review on security issues in vehicular ad hoc network. *Security and Privacy*, 1(5):e39, 2018.

[311] Y. Tian, J. Yuan, and H. Song. Efficient privacy-preserving authentication framework for edge-assisted Internet of Drones. *Journal of Information Security and Applications*, 48:102354, 2019.

[312] A. Tomar and S. Tripathi. Blockchain-assisted authentication and key agreement scheme for fog-based smart grid. *Cluster Computing*, 25(1):451–468, 2022.

[313] M. Torky and A. E. Hassanein. Integrating blockchain and the internet of things in precision agriculture: Analysis, opportunities, and challenges. *Computers and Electronics in Agriculture*, 178:105476, 2020.

[314] T. H. Tran, H. L. Pham, and Y. Nakashima. A High-Performance Multimem SHA-256 Accelerator for Society 5.0. *IEEE Access*, 9:39182–39192, 2021.

[315] F. Tschorsch and B. Scheuermann. Bitcoin and Beyond: A Technical Survey on Decentralized Digital Currencies. *IEEE Communications Surveys Tutorials*, 18(3):2084–2123, 2016.

[316] J. Tu, J. Zhang, S. Chen, T. Weise, and L. Zou. An Improved Retrieval Method for Multi-Transaction Mode Consortium Blockchain. *Electronics*, 9(2), 2020.

[317] A. Tzounis, N. Katsoulas, T. Bartzanas, and C. Kittas. Internet of Things in agriculture, recent advances and future challenges. *Biosystems Engineering*, 164:31 – 48, 2017.

[318] S. Umamaheswari. Internet of Things Practices for Smart Agriculture. In *Smart Farming Technologies for Sustainable Agricultural Development*, pages 67–92. IGI Global, 2019.

[319] J. Unar, W. C. Seng, and A. Abbasi. A Review of Biometric technology along with trends and prospects. *Pattern recognition*, 47(8):2673–2688, 2014.

[320] A. Vasudevan, D. A. Kumar, and N. S. Bhuvaneswari. Precision farming using unmanned aerial and ground vehicles. In *IEEE Technological Innovations in ICT for Agriculture and Rural Development (TIAR)*, pages 146–150, 2016.

[321] L. Vidyashree and B. M. Suresha. Methodology to secure agricultural data in iot. In *Emerging Technologies in Data Mining and Information Security*, pages 129–139, Singapore, 2019. Springer Singapore.

[322] D. von Oheimb. The High-Level Protocol Specification Language HLPSL developed in the EU project AVISPA. In *3rd Workshop of Applied Semantic (APPSEM'05)*, pages 1–17, Frauenchiemsee, Germany, 2005.

[323] J. Vora, A. Nayyar, S. Tanwar, S. Tyagi, N. Kumar, M. S. Obaidat, and J. J. P. C. Rodrigues. BHEEM: A Blockchain-Based Framework for Securing Electronic Health Records. In *37th IEEE Globecom Workshops (GC Worksohps)*, pages 1–6, Abu Dhabi, United Arab Emirates, 2018.

[324] J. Vora, S. Tanwar, S. Tyagi, N. Kumar, and J. J. P. C. Rodrigues. Home-based exercise system for patients using IoT enabled smart speaker. In *19th IEEE International Conference on e-Health Networking, Applications and Services (Healthcom)*, pages 1–6, Dalian, China, 2017.

[325] Q. Vu, M. Raković, V. Delic, and A. Ronzhin. Trends in Development of UAV-UGV Cooperation Approaches in Precision Agriculture. In *Interactive Collaborative Robotics*, pages 213–221, Cham, 2018. Springer International Publishing.

[326] D. Wang, H. Cheng, P. Wang, X. Huang, and G. Jian. Zipf's Law in Passwords. *IEEE Transactions on Information Forensics and Security*, 12(11):2776–2791, 2017.

[327] D. Wang, D. He, P. Wang, and C. Chu. Anonymous Two-Factor Authentication in Distributed Systems: Certain Goals Are Beyond Attainment. *IEEE Transactions on Dependable and Secure Computing*, 12(4):428–442, 2015.

[328] L. Wang, L. Xu, Z. Zheng, S. Liu, X. Li, L. Cao, J. Li, and C. Sun. Smart Contract-Based Agricultural Food Supply Chain Traceability. *IEEE Access*, 9:9296–9307, 2021.

[329] W. Wang, D. T. Hoang, P. Hu, Z. Xiong, D. Niyato, P. Wang, Y. Wen, and D. I. Kim. A Survey on Consensus Mechanisms and Mining Strategy Management in Blockchain Networks. *IEEE Access*, 7:22328–22370, 2019.

[330] L. C. Washington. *Elliptic curves: number theory and cryptography*. Chapman and Hall/CRC, 2008.

[331] M. Wazid, A. K. Das, N. Kumar, V. Odelu, A. Goutham Reddy, K. Park, and Y. Park. Design of Lightweight Authentication and Key Agreement Protocol for Vehicular Ad Hoc Networks. *IEEE Access*, 5:14966–14980, 2017.

[332] M. Wazid, A. K. Das, N. Kumar, and J. J. P. C. Rodrigues. Secure Three-Factor User Authentication Scheme for Renewable-Energy-Based Smart Grid Environment. *IEEE Transactions on Industrial Informatics*, 13(6):3144–3153, 2017.

[333] M. Wazid, A. K. Das, N. Kumar, and A. V. Vasilakos. Design of secure key management and user authentication scheme for fog computing services. *Future Generation Computer Systems*, 91:475 – 492, 2019.

[334] M. Wazid, A. K. Das, S. Kumari, X. Li, and F. Wu. Design of an efficient and provably secure anonymity preserving three-factor user authentication and key agreement scheme for TMIS. *Security and Communication Networks*, 9(13):1983–2001, 2016.

[335] M. Wazid, A. K. Das, V. Odelu, N. Kumar, and W. Susilo. Secure Remote User Authenticated Key Establishment Protocol for Smart Home Environment. *IEEE Transactions on Dependable and Secure Computing*, 17(2):391–406, 2020.

[336] J. West. A Prediction Model Framework for Cyber-Attacks to Precision Agriculture Technologies. *Journal of Agricultural & Food Information*, 19(4):307–330, 2018.

[337] M. Window. Security in precision agriculture: Vulnerabilities and risks of agricultural systems, 2019. https://www.diva-portal.org/smash/get/diva2:1322203/FULLTEXT02.

[338] A. D. Wood and J. A. Stankovic. Denial of service in sensor networks. *Computer*, 35(10):54–62, 2002.

[339] G. Wood et al. Ethereum: A Secure Decentralised Generalised Transaction Ledger. *Ethereum project yellow paper*, 151(2014):1–32, 2014.

[340] H. Wu and C. Tsai. An intelligent agriculture network security system based on private blockchains. *Journal of Communications and Networks*, 21(5):503–508, 2019.

[341] L. Wu, J. Wang, K.-K. R. Choo, and D. He. Secure Key Agreement and Key Protection for Mobile Device User Authentication. *IEEE Transactions on Information Forensics and Security*, 14(2):319–330, 2019.

[342] L. Wu, J. Wang, S. Zeadally, and D. He. Anonymous and Efficient Message Authentication Scheme for Smart Grid. *Security and Communication Networks*, 2019:4836016:1–4836016:12, 2019.

[343] L.-X. Yang, K. Huang, X. Yang, Y. Zhang, Y. Xiang, and Y. Y. Tang. Defense Against Advanced Persistent Threat Through Data Backup and Recovery. *IEEE Transactions on Network Science and Engineering*, 8(3):2001–2013, 2021.

[344] X. Yang, L. Shu, J. Chen, M. A. Ferrag, J. Wu, E. Nurellari, and K. Huang. A Survey on Smart Agriculture: Development Modes, Technologies, and Security and Privacy Challenges. *IEEE/CAA Journal of Automatica Sinica*, 8(2):273–302, 2021.

[345] Y. Yao, X. Chang, J. Misic, V. B. Misic, and L. Li. BLA: Blockchain-Assisted Lightweight Anonymous Authentication for Distributed Vehicular Fog Services. *IEEE Internet of Things Journal*, 6(2):3775–3784, 2019.

[346] S. Yousefi, H. Karimipour, and F. Derakhshan. Data Aggregation Mechanisms on the Internet of Things: A Systematic Literature Review. *Internet of Things*, 15:100427, 2021.

[347] J. Yuan, W. Liu, J. Wang, J. Shi, and L. Miao. An efficient framework for data aggregation in smart agriculture. *Concurrency and Computation: Practice and Experience*, 33(10):e6160, 2021.

[348] H. Zhang, J. Wang, and Y. Ding. Blockchain-based decentralized and secure keyless signature scheme for smart grid. *Energy*, 180:955–967, 2019.

[349] N. Zhang, M. Wang, and N. Wang. Precision agriculture–a worldwide overview. *Computers and Electronics in Agriculture*, 36(2):113–132, 2002.

[350] R. Zhang, R. Xue, and L. Liu. Security and Privacy on Blockchain. *ACM Comput. Surv.*, 52(3), 2019.

[351] S. Zhang and J. Lee. A Group Signature and Authentication Scheme for Blockchain-Based Mobile-Edge Computing. *IEEE Internet of Things Journal*, 7(5):4557–4565, 2020.

[352] Y. Zhang. The Role of Precision Agriculture. *Resource Magazine*, 26(6):9–9, 2019.

[353] M. Zheng, S. Zhang, Y. Zhang, and B. Hu. Construct Food Safety Traceability System for People's Health Under the Internet of Things and Big Data. *IEEE Access*, 9:70571–70583, 2021.

[354] Z. Zheng, S. Xie, H.-N. Dai, X. Chen, and H. Wang. Blockchain challenges and opportunities: A survey. *International Journal of Web and Grid Services*, 14(4):352–375, 2018.

[355] B. Zhou, H. Li, and L. Xu. An Authentication Scheme Using Identity-based Encryption Blockchain. In *IEEE Symposium on Computers and Communications (ISCC)*, pages 556–561, Natal, Brazil, 2018.

[356] M. Zhou, Y. Zheng, Y. Guan, L. Peng, and R. Lu. Efficient and privacy-preserving range-max query in fog-based agricultural IoT. *Peer-to-Peer Networking and Applications*, 14:2156–2170, 2021.

Publications from this Thesis Work

Journal Papers

J1. Anusha Vangala, Ashok Kumar Das, Ankush Mitra, Sajal Kumar Das, and YoungHo Park. "Blockchain-Enabled Authenticated Key Agreement Scheme for Mobile Vehicles-Assisted Precision Agricultural IoT Networks" in ***IEEE Transactions on Information Forensics and Security***, Vol. 18, pp. 904-919, 2022, DOI: 10.1109/TIFS.2022.3231121. (2021 SCI Impact Factor: 7.231)

J2. Anusha Vangala, Anil Kumar Sutrala, Ashok Kumar Das, and Minho Jo. "Smart Contract-Based Blockchain-Envisioned Authentication Scheme for Smart Farming," in ***IEEE Internet of Things Journal***, Vol. 8, No. 13, pp. 10792-10806, July 2021, DOI: 10.1109/JIOT.2021.3050676. (2021 SCI Impact Factor: 10.238)

J3. Basudeb Bera, Anusha Vangala, Ashok Kumar Das, Pascal Lorenz, and Muhammad Khurram Khan. "Private blockchain-envisioned drones-assisted authentication scheme in IoT-enabled agricultural environment," in ***Computer Standards & Interfaces*** **(Elsevier)**, Vol. 80, Article No. 103567, pp. 1-18, 2022, DOI: 10.1016/j.csi.2021.103567. (2021 SCI Impact Factor: 3.721)

J4. Anusha Vangala, Ashok Kumar Das, Vinay Chamola, Valery Korotaev, and Joel J. P. C. Rodrigues. "Security in IoT-enabled Smart Agriculture: Architecture, Security Solutions and Challenges," in ***Cluster Computing*** **(Springer)**, 2022, DOI: 10.1007/s10586-022-03566-7. (2021 SCI Impact Factor: 2.303)

J5. Anusha Vangala, Ashok Kumar Das, Neeraj Kumar, and Mamoun Alazab. "Smart Secure Sensing for IoT-Based Agriculture: Blockchain Perspective," in ***IEEE Sensors Journal***, Vol. 21, No. 16, pp. 17591-17607, August 2021, DOI: 10.1109/JSEN.2020.3009382. (2021 SCI Impact Factor: 4.325)

J6. Anusha Vangala, Ashok Kumar Das, and Jong-Hyouk Lee. "Provably secure signature-based anonymous user authentication protocol in an Internet of Things-enabled intelligent precision agricultural environment," in ***Concurrency and Computation: Practice and Experience*** **(Wiley)**, 2021, pp. e6187, DOI: 10.1002/cpe.6187. (2021 SCI Impact Factor: 1.831)

Other Publications

Journal Papers

J1. Anusha Vangala, Basudeb Bera, Sourav Saha, Ashok Kumar Das, Neeraj Kumar, and YoungHo Park. "Blockchain-Enabled Certificate-Based Authentication for Vehicle Accident Detection and Notification in Intelligent Transportation Systems," in ***IEEE Sensors Journal***, Vol. 21, No. 14, pp. 15824-15838, July 2021, DOI: 10.1109/JSEN.2020.3009382. (2021 SCI Impact Factor: 4.325)

J2. Anusha Vangala, Ashok Kumar Das, YoungHo Park, and Sajjad Shaukat Jamal. "Blockchain-based Robust Data Security Scheme in IoT-enabled Smart Home Envisioned Ubiquitous Computing Environment," in ***CMC-Computers, Materials & Continua***, Vol. 72, No. 2, pp. 3549-3570, 2022, DOI:10.32604/cmc.2022.025660. (2021 SCI Impact Factor: 3.860)

Conference Papers

C1. Anusha Vangala, Sandip Roy, and Ashok Kumar Das. Blockchain-based Lightweight Authentication Protocol for IoT-Enabled Smart Agriculture. In ***IEEE International Conference on Cyber-physical Social Intelligence (ICCSI'22)***, Nanjing, China, 11-14 November 2022. pp. 110-115, DOI: 10.1109/ICCSI55536.2022.9970603.

C2. Anusha Vangala and Ashok Kumar Das. Privacy-preserving Blockchain-based Authentication in Smart Energy Systems. In ***20th ACM Conference on Embedded Networked Sensor Systems (SenSys'22): Fourth ACM International Workshop on Blockchain-enabled Networked Sensor Systems (BlockSys'22)***, Boston, MA, USA, 6-9 November 2022.

Book Chapters

BC1. Anusha Vangala, Ashok Kumar Das, Volkan Dedeoglu, and Raja Jurdak. "Security and Privacy of Cyber Physical Systems," in ***Chapter 6, Cyber-Physical Systems for Industrial Transformation: Fundamentals, Standards, and Protocols***, editors G. Manogaran, N. E. M. Khalifa, M. Loey, and M. H. N. Taha, 2022, CRC Press, Taylor & Francis, USA.

BC2. Anusha Vangala and Ashok Kumar Das. "Blockchain-based Fog Computing," in ***Chapter 3, Security Issues in Fog Computing from 5G to 6G: Architectures, Applications and Solutions***, editors C. Bhatt, Y. Wu, S. Harous, and M. Villari, Springer, pages 31-58, 2022.