Computer Standards & Interfaces 80 (2022) 103567

Contents lists available at ScienceDirect

Computer Standards & Interfaces

journal homepage: www.elsevier.com/locate/csi

Private blockchain-envisioned drones-assisted authentication scheme in IoT-enabled agricultural environment

Basudeb Bera [a], Anusha Vangala [a], Ashok Kumar Das [*,a], Pascal Lorenz [b], Muhammad Khurram Khan [c]

[a] Center for Security, Theory and Algorithmic Research, International Institute of Information Technology, Hyderabad 500 032, India
[b] University of Haute Alsace, France
[c] College of Computer & Information Sciences, King Saud University, Riyadh, Saudi Arabia

ARTICLE INFO

Keywords:
Intelligent precision agriculture (IPA)
Internet of things (IoT)
Drones
Blockchain
Authentication
Security

ABSTRACT

In an Intelligent Precision Agriculture (IPA), several Internet of Things (IoT) smart devices and drones can be deployed to monitor an agricultural environment. The drones can be further utilized to collect the data from smart devices and send 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. For this goal, a new authentication and key management scheme for IoT-enabled IPA, called AKMS-AgriIoT, has been put forward with the private blockchain-based solution. 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 in the private blockchain center. A detailed security analysis and comparative study reveal that the proposed AKMS-AgriIoT supports better security, and provides more functionality features, less communication costs and comparable computation costs as compared to other relevant schemes. In addition, the blockchain-based implementation on the proposed AKMS-AgriIoT has been also carried out.

1. Introduction

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 a 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) using IoT-enabled drones (commonly known as unmanned aerial vehicles (UAVs) or uncrewed aerial vehicles) being deployed to monitor and control the approach to smart farming [1–3].

IPA is expected to have the following properties according to Rubio et al. [4]: 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 favor of the circumstances in ROI; 5) actuation technologies: the decided action needs to be executed either in real-time or deferred-time; 6) interoperability of devices: it involves a network of diverse devices interacting to collect and analyze data.

PA is treated as a mechanism related to farm management in which the information technology (IT) plays an important role for 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. In smart farming, several Internet of Things (IoT) smart devices can be deployed to monitor the agricultural environment. The drones can be further utilized to collect the data sensed by the IoT smart devices and even sometimes they can collect directly the information from the specific flying zones. The data are then sent to the Ground Station Server (*GSS*) by the drones. However, insecure communication among the

* Corresponding author.
E-mail addresses: basudeb.bera@research.iiit.ac.in (B. Bera), anusha.vangala@research.iiit.ac.in (A. Vangala), iitkgp.akdas@gmail.com, ashok.das@iiit.ac.in (A.K. Das), lorenz@ieee.org (P. Lorenz), mkhurram@ksu.edu.sa (M.K. Khan).

https://doi.org/10.1016/j.csi.2021.103567
Received 18 February 2021; Received in revised form 4 August 2021; Accepted 5 August 2021
Available online 9 August 2021

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 drones capture attacks. Apart from these, we need anonymity and untraceability properties that need to be highly maintained 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 widely-accepted Practical Byzantine Fault Tolerance (PBFT) algorithm to verify and add the blocks.

1.1. Network model

The architecture developed for the IoT-enabled drones-assisted agriculture environment depicted in Fig. 1 is divided into m flying zones FZ_p $(p = 1, 2, \ldots, m)$, with each zone consisting of a number of IoT smart devices SD_j $(j = 1, 2, \ldots, n_{sd})$ and they are monitored by a drone DR_i $(i = 1, 2, \ldots, n_{dr})$, where n_{sd} and n_{dr} denote the number of deployed smart devices and drones, respectively. The Control Room (CR) does the registration of the Ground Station Server (GSS), drones and also smart devices prior to their 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 established session keys. In a similar way, the drones and the *GSS* involve in 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 verifying by other cloud servers in a Peer-to-Peer (P2P) cloud servers network using consensus algorithm. Finally, the blocks are stored by the cloud servers in decentralized manner in the blockchain center.

1.2. Threat model

We adopt the widely-accepted "Dolev-Yao (DY) threat model" [5] along with "Canetti and Krawczyk's model (CK-adversary model)" [6] in the proposed AKMS-AgriIoT. Thus, an adversary $\mathscr{A}$ not only can inject the malicious information, but also can modify or delete the message contents in between the communication. Moreover, $\mathscr{A}$ can compromise secret credentials, secret keys and even session states "if those information are available in insecure memory of the participants through the session hijacking attack". In addition, the end-point communicating parties (drones and IoT smart devices) are not contemplated as trustworthy entities in the network. Since an agriculture field can not be monitored in 24 × 7 hours, $\mathscr{A}$ may physically compromise some drones as well as smart devices. Once a node is physically compromised, all the stored credentials can be pulled out by $\mathscr{A}$ using sophisticated "power analysis attacks" [7] and these information can be utilized for "mounting other attacks, such as impersonation attacks".

1.3. Motivation

The usage of blockchain technology in the network model shown in Fig. 1 is prominent, because it enables storage of sensitive data related to the agricultural crop management to be stored securely. This data is collected into transactions by the *GSS*. Each transaction is encrypted with the public key of the *GSS* to ensure that only the *GSS* will be able to access the contents of the transactions and later digitally signed by the *GSS* that confirms the transactions are being assembled for the block only by the *GSS*. This specifically ensures secure access to the transaction contents, along with the confidence that except for the *GSS* no

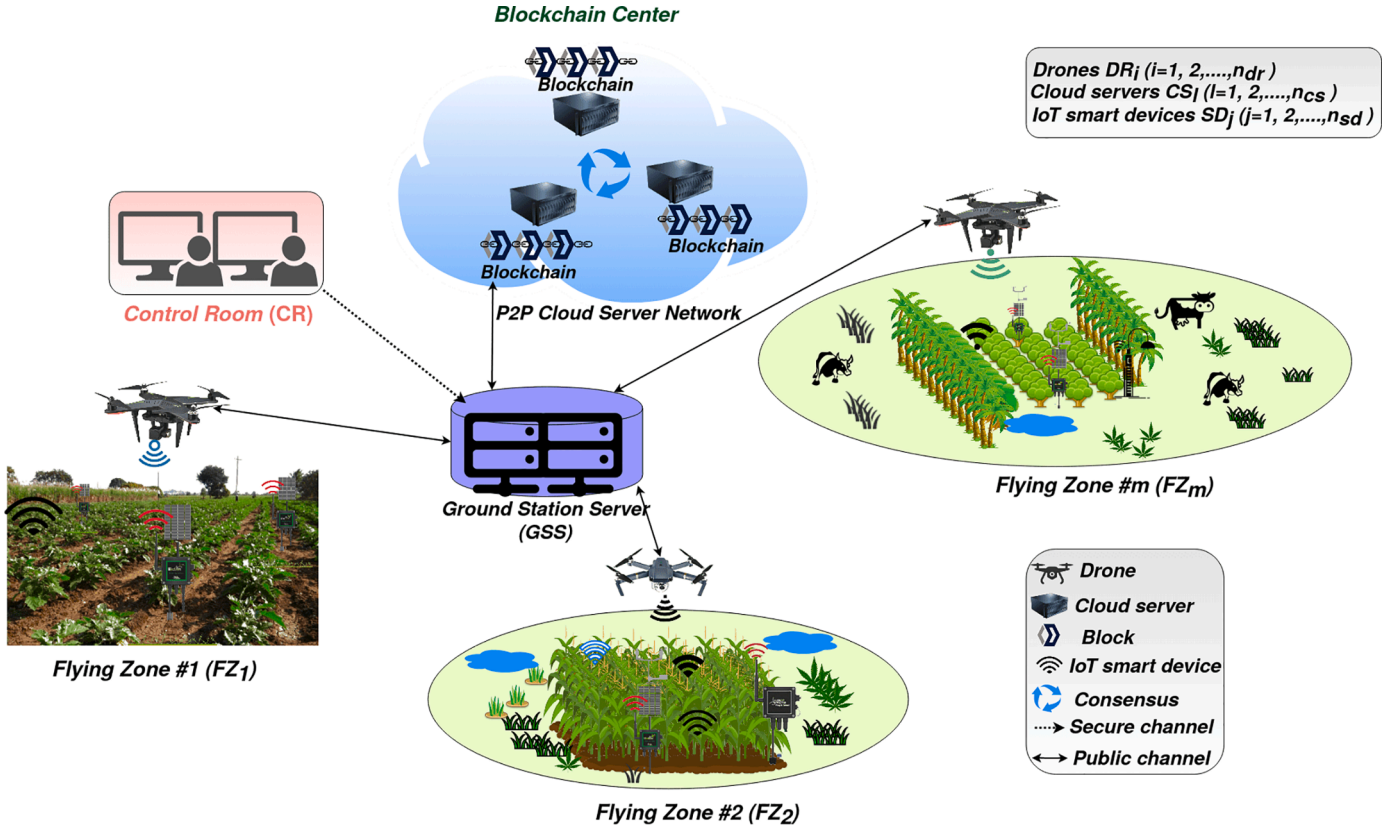

Fig. 1. Blockchain-envisioned IoT-enabled agricultural environment using drones.

other entity can tamper with the transactions and any modifications can be traced back to the *GSS*. After this collection of transactions is received by a cloud server (CS), a consensus is achieved on the correctness of the contents using a consensus algorithm with the help of other *CS* and only then a block is created to be added to the blockchain. Once added to the blockchain, it can only be accessed but not removed or modified thus enabling permanence of information. Anonymity is achieved as except for the *GSS*, no other entity can trace the data back to the sensors which sent it. It is shown in the cost analysis that the presented proposed solution is very efficient in terms of communication cost and the security attributes it supports despite only comparable increase in computation cost.

The specific solutions based on blockchain technology are very practical in the precision agriculture scenario as it can help to keep extremely sensitive data about the agriculture related to the growth and health of harvest, and the obtained produce with persistence, auditability and anonymity. The data from the agricultural fields needs 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 affects the farmers' livelihood. Therefore, the solution presented in this paper is 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.

Ge et al. [8] suggests that the agricultural sector is prone to the risk of fraud due to packaged food and beverage companies increasingly needing proper certification to their products, which can directly affect the health of its consumers. This has lead to the identification for the need of transparency and trust in the agricultural sector. Blockchain aims to eliminate the participation of third parties in the audits and turning 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 [9] have 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. Akram et al. [10] proposed that the usage of blockchain technology will be invaluable in various areas, 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 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. [11] 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.

The network model for the proposed scheme shows the usage of drones in an IoT-enabled agricultural field. A drone is an unmanned aerial vehicle that is capable of performing several tasks such as monitoring field conditions, plantation of seeds, spraying of pesticides and fertilizers, pollination and irrigation. Tripicchio et al. [12], Stehr [13], Puri et al. [14], Kulbacki et al. [15], Ayamga et al. [16] and Deon et al. [17] emphasize the usage of drones for these tasks that can aid in managing the fields well regardless of the availability of manpower or livestock for the tasks. Moreover, the "Food and Agriculture Organization of the United Nations (FAO) in collaboration with International Telecommunication Union (ITU)" [18] presents various case studies for using drones in agriculture. With these motivations, in this paper, we aim to design a new authentication and key management scheme for IoT-enabled IPA (AKMS-AgriloT) which uses the IoT-based agricultural environment using drones.

1.4. Research contributions

The following are the main contributions towards this work:

- We design a "new authentication and key management scheme for IoT-enabled IPA, called AKMS-AgriIoT". AKMS-AgriIoT is supported with 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 and these are added after verification by "other cloud servers in the P2P CS network" using 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)" [19], 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)" [20] 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 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.

1.5. Paper outline

In Section 2, we discuss the related work on IoT-enabled authentication protocols and their limitations. While Section 3 introduces a new authenticated key management scheme for IoT-enabled drones-assisted agricultural environment (AKMS-AgriIoT), Sections 4 and 5 provide its detailed security analysis. The experimental results for various cryptographic primitives using MIRACL are demonstrated in Section 6. A detailed comparative study on various parameters on the proposed AKMS-AgriIoT and other relevant authentication schemes is given in Section 7. The blockchain implementation of the proposed scheme is then provided in Section 8. Finally, the concluding remarks are given in Section 9.

2. Related work

Authentication and access control are two primary important security services that are applied to secure various networking environments, such as wireless sensor networks, IoT, Internet of Drones (IoD) and agriculture monitoring [21–36].

Wang et al. [33] provided a comprehensive measurement on user authentication schemes proposed in wireless sensor networks (WSNs) and IoT. They observed that it leads to an unsatisfactory "break--fix-break-fix" cycle. They designed a "systematical evaluation criteria" for user authentication schemes that need to be assessed objectively. In particular, they cryptanalyzed the schemes proposed by Wu et al. [37] and Srinivas et al. [38] in order to show the challenges as well as difficulties faced in designing a strong robust user authentication scheme. Wu et al.'s scheme [37] fails to protect "smart card loss attack", "user impersonation attack", and also "user anonymity violation attack". On

the other side, Srinivas et al.'s scheme [38] does not protect "smart card loss attack", "physical sensing device capture attack", and also "user untraceability violation attack". Their important observations reveal that most user authentication schemes proposed in the area of IoT and WSNs are not ideal for practical applications. This certainly gives us a pointer that we need a robust user authentication scheme.

Ali et al. [36] designed a "remote user authentication protocol" for wireless sensor networking environment based agriculture monitoring system. It allows a user to securely access the real-time data from accessed sensing devices that are deployed in the network. Though their scheme is lightweight, it suffers from several security pitfalls, such as "privileged-insider attack" and "Ephemeral Secret Leakage (ESL)" attacks, and it fails to support anonymity and untraceability issues.

Tian et al. [34] 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 ESL attack.

Tai et al. [39] also 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.

More recent work by Sadhukhan et al. [41] was developed to provide an authentication scheme to allow a registered user to authenticate a sensor and vice-versa via the gateway node by creating a session key using elliptic curve cryptographic and symmetric key encryption methods. Even though this scheme is resistant to replay and man-in-the-middle attacks, it is vulnerable against user impersonation, ESL and privileged insider attacks, and does not support dynamic node addition, user device revocation, user biometric and password change phases. Shuai et al. [42] is also another recently developed authentication scheme relying on public-key based Rabin cryptosystem that allows a user mobile device and a sensor to mutually authenticate each other via the industrial management gateway. Similar to Sadhukhan et al. as discussed above, this scheme is resistant to man-in-the-middle and replay attacks, but it stands unguarded against user impersonation, ESL and privileged insider attacks and is without any supporting features of user device revocation and biometric change phases.

In recent years, the blockchain-based solution becomes a promising technique to make the data transparent and immutable, and also to store the data in decentralized way so that a single-server failure issue will not arise [44–48]. Though Wu and Tsai [43] proposed an "intelligent agriculture network security system based on private blockchains", their scheme requires high computation due to involvement of costly bilinear pairing operations. The architecture proposed by Wu and Tsai [43] involves the use of dark web with blockchain along with 4G communication, which will not only increase the overheads but also not logically apply for the scenario discussed in the network model. Almadhoun et al. [40] also proposed a blockchain-based scheme that is based on a hierarchical architecture with fog nodes, and hence, it will not be applicable for the network model as discussed in our scheme.

Finally, in Table 1, we summarize the cryptographic techniques, characteristics and limitations of the discussed existing state of art authentication schemes that can be deployed in an IoT environment.

Table 1
Cryptographic techniques, characteristics and limitations of existing authentication schemes in IoT deployment.

Scheme	Year	Cryptographic Techniques	Characteristics	Limitations
Wu et al. [37]	2017	* Elliptic curve cryptography * Symmetric encryption * Hash functions	* User authentication * Session key agreement	* Vulnerable to smart card loss and user impersonation attacks, and user anonymity violation * Does not support blockchain solution
Srinivas et al. [38]	2017	* Biohashing	* User authentication * Session key agreement	* Vulnerable to smart card loss attack, physical sensing device capture attack and user untraceability violation attack * Does not support blockchain solution
Tai et al. [39]	2017	* Hash functions * Smart card	* Mutual authentication * Session key agreement	* Vulnerable to privileged-insider, malicious IoT device deployment, ESL and physical device capture attacks * Does not support blockchain solution
Ali et al. [36]	2018	* Fuzzy extractor * Smart card * Symmetric encryption	* Lightweight user authentication * Session key agreement	* Does not support blockchain solution
Almadhoun et al. [40]	2018	* Hashing * Asymmetric cryptography	* User authentication * Fog computing * Supports blockchain solution	* Does not meet in most IoT scenarios * High communication cost
Tian et al. [34]	2019	* Modular exponentiations * Hash functions *RSA-based digital signature	* Mutual authentication * Session key establishment	* Vulnerable to ESL attack under the CK-adversary model * Does not support blockchain solution
Sadhukhan et al. [41]	2020	* Symmetric encryption * Elliptic curve cryptography	* User authentication * Session key agreement	* Vulnerable to user impersonation, ESL and privileged insider attacks * Does not support user device revocation, dynamic node addition, user biometric and password change phases * Does not support blockchain
Shuai et al. [42]	2020	* Rabin cryptosystem	* User authentication * Session key agreement	* Vulnerable to user impersonation, ESL and privileged insider

(continued on next page)

3. The proposed 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 Fig. 1. We apply the current timestamps and random nonces to achieve strong replay

Table 1 (*continued*)

Scheme	Year	Cryptographic Techniques	Characteristics	Limitations
				attacks * Does not support user device revocation, biometric change phases * Does not support blockchain
Wu and Tsai [43]	2020	* Symmetric Encryption * HMAC (hashed message authentication code) * Bilinear pairings	* Mutual authentication * Session key agreement * Support blockchain solution	* Expensive computations
Proposed (AKMS-AgriIoT)	2021	* Elliptic curve cryptography * Hashing * t-degree symmetric bivariate polynomial	* Mutual authentication * Session key agreement * Support blockchain solution	* Implementation in real-world environment is needed as future research work

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 [49,50]. 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 [51] 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 2 are utilized for analyzing and discussing the proposed AKMS-AgriIoT.

It is worth noticing that the fully trusted control room (CR) only involves 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 "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 registration of various network entities instead of the cloud servers in this paper.

In this work, the blockchain technology has been adapted in the proposed system in order to support the authentication and key management designed for providing an effective way to store data (information) in form of transactions. The blockchain technology provides a 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 immutability property of the blockchain. The credentials are verified thoroughly using an appropriate consensus algorithm before being permanently stored into the blockchain. Once stored into the blockchain, the stored credentials cannot be modified in any possible way; thereby increasing the confidence that the details cannot be used by any attacker to launch 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 decentralized manner and replicated among multiple cloud servers.

Table 2
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: $x{\cdot}G = G + G + \cdots + G(xtimes)$
$P + Q$	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 keys, respectively
k_{GSS}, Pub_{GSS}	GSS's private and public keys, respectively
DR_i, ID_{DR_i}	ith 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 keys, 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	lth 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}	jth 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 $a_{ij} \in Z_q = \{0,1,2,\ldots,q-1\}$
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	mth flying zone in an agricultural environment

The detailed description of each phase is given below.

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 $u, v \in Z_q = \{0, 1, 2, \ldots, q-1\}$ 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 [52]), the "elliptic curve digital signature algorithm (ECDSA)" [53], and the widely-accepted PBFT consensus algorithm [51].
- 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.

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 which is carried by the CR and it is only involve during this process.

3.2.1. 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" [54,55]. Next, the CR computes a polynomial share $f(PID_{GSS}, y)$ using the pseudo-identity $PID_{GSS} = h(ID_{GSS} \parallel mk_{CR} \parallel RTS_{GSS})$, 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 $Cert_{GSS} = r_{GSS} + h(PID_{GSS} \parallel ID_{CR} \parallel R_{GSS} \parallel Pub_{CR}) * mk_{CR} \pmod q$. 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 Fig. 2.

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

Fig. 2. Summary of GSS registration phase.

3.2.2. 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 $PID_{DR_i} = h(ID_{DR_i} \parallel mk_{CR} \parallel RTS_{DR_i})$ 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 $Cert_{DR_i} = r_{DR_i} + h(PID_{DR_i} \parallel ID_{CR} \parallel R_{DR_i} \parallel Pub_{CR}) * mk_{CR} \pmod q$, 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 Fig. 3.

3.2.3. 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 $PID_{SD_j} = h(ID_{SD_j} \parallel mk_{CR} \parallel RTS_{SD_j})$ 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 Fig. 4.

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 $A_{SD_j} = h(a_{SD_j} \parallel TID_{SD_j} \parallel PID_{SD_j} \parallel k_{SD_j} \parallel TS_{SD_j}) \cdot G$, and generates the signature on a_{SD_j} as $Sign_{SD_j} = h(a_{SD_j} \parallel$

Control Room (CR)	Drone (DR_i)
Generate ID_{DR_i}, RTS_{DR_i}.	
Generate a random secret $r_{DR_i} \in Z_q^*$.	
Calculate $R_{DR_i} = r_{DR_i} \cdot G$,	
$PID_{DR_i} = h(ID_{DR_i} \parallel mk_{CR} \parallel RTS_{DR_i})$.	
Generate TID_{DR_i}. Create certificate as	
$Cert_{DR_i} = r_{DR_i} + h(PID_{DR_i} \parallel ID_{DR_i}$	
$\parallel R_{DR_i} \parallel Pub_{CR}) * mk_{CR} \pmod q$.	
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)
$\{(TID_{DR_i}, PID_{DR_i})\} \xrightarrow{}$	
(via secure channel)	

Fig. 3. Summary of drone registration phase.

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)
$\{(TID_{SD_j}, PID_{SD_j})\}$ $\xrightarrow{\text{(via secure channel)}}$	

Fig. 4. Summary of smart device registration phase.

$TID_{SD_j} \| PID_{SD_j} \| k_{SD_j} \| TS_{SD_j}) + h(Pub_{SD_j} \| Pub_{DR_i} \| Pub_{GSS} \| TS_{SD_j}) * k_{SD_j} \pmod q$. 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 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 $TID^*_{SD_j} = TID^{new}_{SD_j} \oplus h(TID_{SD_j} \| SK_{DR_i,SD_j} \| Sign_{DR_{i_1}} \| TS_{DR_{i_1}})$. 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 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 retrieve $TID^{new}_{SD_j}$ 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} $(= SK_{DR_i,SD_j})$.

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 $(l = 1, 2, \ldots, m)$. 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 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 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 passes 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 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 illustrates briefly both the phases described in Sections 3.3 and 3.4.

3.5. Block creation, verification and addition in BC

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 are 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 existing private blockchain by the *CS*, the block verification is executed through the consensus algorithm. In this paper, we use the "Practical Byzantine Fault Tolerance (PBFT)" consensus algorithm [51].

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 3.4. After that the *GSS* makes several transactions, say n_t transactions $Tx_1, Tx_2, \ldots, Tx_{n_t}$ for a block $Block_m$, and then encrypts all these n_t

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 $Sign_{SD_j}$. $Msg_{SD_1} = \{TID_{SD_j}, A_{SD_j}, Sign_{SD_j}, TS_{SD_j}\}$ (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.
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 $TID_{SD_j}^{new}$.	Calculate $TID_{SD_j}^* = TID_{SD_j}^{new} \oplus h(TID_{SD_j} \|\|SK_{DR_i,SD_j} \|\|Sign_{DR_{i_1}} \|\|TS_{DR_{i_1}})$. $Msg_{SD_2} = \{TID_{SD_j}^*, B_{DR_i}, Sign_{DR_{i_1}}, TS_{DR_{i_1}}\}$ (via open channel) 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 $Sign_{x_{GSS}}$. $Msg_{GD1} = \{TID_{DR_i}, X_{GSS}, Cert_{GSS}, Sign_{x_{GSS}}, PID_{GSS}^*, TS_{GSS}\}$ (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 $TID_{DR_i}^* = TID_{DR_i}^{new} \oplus h(TID_{DR_i} \|\|f(PID_{DR_i}, PID_{GSS}) \|\|SK_{DR_i,GSS} \|\|TS_{DR_i})$. $Msg_{GD2} = \{TID_{DR_i}^*, Cert_{DR_i}, Y_{DR_i}, Sign_{DR_i}, TS_{DR_i}\}$ (via open channel) Update TID_{DR_i} with $TID_{DR_i}^{new}$.
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 $TID_{DR_i}^{new}$.	

Both GSS and DR_i share the same secret key SK_{GSS,DR_i} $(= SK_{DR_i,GSS})$

Fig. 5. Summary of authentication and key management phases.

transaction using its own public key Pub_{GSS} as $\{E_{Pub_{GSS}}(Tx_1), E_{Pub_{GSS}}(Tx_2), \ldots, E_{Pub_{GSS}}(Tx_{n_t})\}$.

- 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 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 $Msg_{GC} = \{(E_{Pub_{GSS}}(Tx_i)|i = 1, 2, \ldots, n_t), ECDSA.sig_{Block_m}\}$ 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 Fig. 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 [56]. The consensus algorithm (PBFT) will be then executed and its detailed description is shown in Algorithm 1. 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*.

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	$CBHash_m$

Fig. 6. Structure of a block $Block_m$ based on transactions.

Algorithm 1 is 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 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 using the *ECDSA* signature verification algorithm, decrypts *EVTReq* using its

Assume a leader (L) is selected which has a block $Block_m = \{BVer_m, PBH_m, MTR_m, TS_m, OB_m, Pub_{GSS}, \{E_{Pub_{GSS}}(Tx_i) | i = 1, 2, \ldots, n_t\}, ECDSA.sig_{Block_m}, CBHash_m\}$.
L generates a current timestamp TS_{CS_l} for each follower cloud server peer node, say CS_l and starts voting process.
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, $l = 1, 2, \ldots, n_{cs}$ with $L \neq CS_l$.
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, $(l = 1, 2, \ldots, n_{cs}, L \neq CS_l)$.
Assume that message is received from the L by every follower CS_l at time $TS^*_{CS_l}$.
for each follower node CS_l **do**
    Verify signature sig_{VTRreq} using $ECDSA.ver$ algorithm.
    if signature is valid **then**
        Compute $(VTReq, TS_{CS_l}) = D_{k_{CS_l}}[EVTReq]$.
        if $(|TS^*_{CS_l} - TS_{CS_l}| < \Delta T)$ **then**
            Compute the Merkle tree root, say $MTR_m\prime$ on the encrypted transactions present in $Block_m$.
            if $(MTR_m\prime = MTR_m)$ **then**
                Compute block hash $CBHash_m\prime$ on all the fields $\{BVer_m, PBH_m, MTR_m, TS_m, OB_m, Pub_{GSS}, \{E_{Pub_{GSS}}(Tx_i) | i = 1, 2, \ldots, n_t\}, ECDSA.sig_{Block_m}\}$.
                if $(CBHash_m\prime = CBHash_m)$ **then**
                    Send the block validation status $BVStatus$ and vote reply $VTRep$ as $\{E_{Pub_L}(VTRep, BVStatus), sig_{VTRep}\}$ to the leader L, where $sig_{VTRep} = ECDSA.sig_{k_{CS_l}}[E_{Pub_L}(VTRep, BVStatus)]$.
                end if
            end if
        end if
    end if
end for
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 $VVCount = VVCount + 1$.
        end if
    end if
end for
if $(VVCount > 2 * f_{n_{cs}} + 1)$ **then**
    Send the commit response to all followers. /* $f_{n_{cs}}$ is the faulty node (CS) present in the network */
    Add block $Block_m$ to the blockchain.
end if

Algorithm 1. Consensus for block validation and addition.

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 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 into the respective distributed ledgers of the peer nodes.

Finally, the overview of the proposed private blockchain-based drones-assisted authentication and key agreement scheme (AKMS-AgriIoT) in an IoT-enabled agricultural environment has been provided in Fig. 7.

3.6. Dynamic nodes addition phase

This phase allows addition of new IoT smart devices due to their power exhaustion or physical device capturing issue 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 $PID_{SD_j}^{new} = h(ID_{SD_j}^{new} \parallel mk_{CR} \parallel RTS_{SD_j}^{new})$ 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}.
- 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 Fig. 8.

Control Room (*CR*)	Smart Device (SD_j^{new})
Generate $ID_{SD_j}^{new}, RTS_{SD_j}^{new}$.	
Calculate $PID_{SD_j}^{new} = h(ID_{SD_j}^{new} \parallel mk_{CR} \parallel RTS_{SD_j}^{new})$.	
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)
$\{TID_{SD_j}^{new_1}, PID_{SD_j}^{new}\}$ $\xrightarrow{\text{(via secure channel)}}$	

Fig. 8. Summary of dynamic smart device addition phase.

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. [57] made an important observation that the broadly-accepted formal mechanisms for security analysis, such as "random oracle model" and "Burrows-Abadi-Needham (BAN) logic [58]" can not capture the "structural mistakes" in a designed user authentication scheme. As a result, guaranteeing soundness of the designed authentication protocols 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 probability.

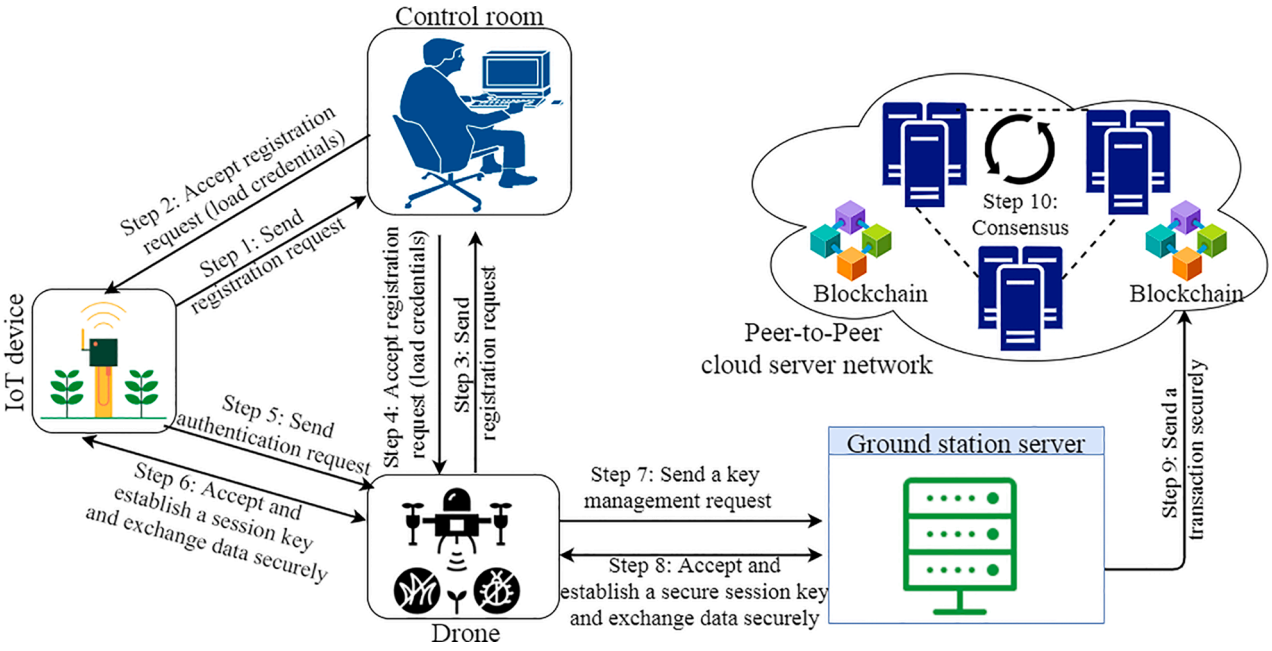

Fig. 7. Overall process diagram of the proposed scheme.

4.1. Formal security analysis under ROR model

The widely-accepted "Real-Or-Random (ROR)" oracle model [59] 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 Fig. 5 for the proposed AKMS-AgriIoT against an adversary $\mathscr{A}$.

In the following, we first briefly provide the description of the ROR model. $\mathscr{A}$ will have access to various queries that are illustrated in Table 3. In addition, as in [60], 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 $\mathscr{A}$.

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 (see Section 3.3), whereas the entities GSS and its associated drone (DR_i) are involved during the key management phase (see Section 3.4). Let $\Pi_{DR_i}^{l_1}$, $\Pi_{GSS}^{l_2}$ and $SD_j^{l_3}$) denote the l_1th, l_2th and l_3th 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 the 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 $\mathscr{A}$ by executing the Reveal(Π^l) query described in Table 3. 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 $\mathscr{A}$ by executing the Reveal(Π^l) query described in Table 3.

Before going to Theorem 1, we define "semantic security" of our proposed AKMS-AgriIoT in Definition 1.

Definition 1. (*Semantic security*) If $Adv_{\mathscr{A}}^{AKMS-AgriIoT}(t_p)$ denote the "advantage of an adversary $\mathscr{A}$ 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,

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

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

Table 3
Queries and their functions.

Query	Description
$Execute(\Pi_{DR_i}^{l_1}, \Pi_{GSS}^{l_2}, SD_j^{l_3})$	This query is executed by $\mathscr{A}$ to intercepted the messages communicated between DR_i, SD_j and GSS
$CorruptDR(\Pi_{DR_i}^{l_1})$	$\mathscr{A}$ executes such a query to extract "secret credentials stored in a compromised drone DR_i"
$CorruptSD(\Pi_{SD_j}^{l_3})$	$\mathscr{A}$ executes such a query to extract "secret credentials stored in a compromised smart device SD_j"
$Reveal(\Pi^l)$	This query is executed by $\mathscr{A}$ 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
$Test(\Pi^l)$	$\mathscr{A}$ 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"

Theorem 1. *Let q_h, $|\mathscr{H}|$, and $Adv_{\mathscr{A}}^{ECDDHP}(t_p)$ 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 $\mathscr{A}$ 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*

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

Proof. The proof of this theorem is followed in a similar way that was done in [26,61–64]. 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 $\mathscr{A}$'s wining $\mathbb{G}ame_i$ by guessing the correct bit c. We define $\mathscr{A}$'s advantage (success probability) by $Adv_{\mathscr{A},\mathbb{G}ame_i}^{AKMS-AgriIoT} = Pr[Succ_{\mathbb{G}ame_i}]$.

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

- $\mathbb{G}ame_0$: Under this game $\mathbb{G}ame_0$, $\mathscr{A}$ executes the real attack against AKMS-AgriIoT under the ROR model. At the beginning of $\mathbb{G}ame_0$, $\mathscr{A}$ picks a random bit c. From the semantic security security defined in Definition 1 of the proposed AKMS-AgriIoT, it follows that

$$Adv_{\mathscr{A}}^{AKMS-AgriIoT}(t_p) = \left|2Adv_{\mathscr{A},\mathbb{G}ame_0}^{AKMS-AgriIoT} - 1\right|. \quad (1)$$

- $\mathbb{G}ame_1$: In this game, $\mathscr{A}$ applies *eavesdropping attack*. $\mathscr{A}$ 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 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 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, $\mathscr{A}$ can simulate the *Test* and *Reveal* queries to validate where the derived session keys are correct or just random values. To derive the session keys, $\mathscr{A}$ needs the short term as well as long term secrets. Since $\mathscr{A}$ can not compromise any one of the long term and short term secrets by intercepting of 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

$$Adv_{\mathscr{A},\mathbb{G}ame_1}^{AKMS-AgriIoT} = Adv_{\mathscr{A},\mathbb{G}ame_0}^{AKMS-AgriIoT}. \quad (2)$$

- $\mathbb{G}ame_2$: In this game, $\mathscr{A}$ executes an active attack. The difference between this game and the previous game $\mathbb{G}ame_1$ is that the simulation of *Hash* quires, *CorruptSD*, *CorruptDR* and solving of ECDDHP are included in $\mathbb{G}ame_2$. Assume that $\mathscr{A}$ 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 $SK_{DR_i,GSS}$ ($= SK_{GSS,DR_i}$), $\mathscr{A}$ 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, $\mathscr{A}$ 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 $\mathscr{A}$ needs to solve ECDDHP too in polynomial time (t_p). Thus, $\mathscr{A}$'s advantage (success probability) of solving ECDDHP in time t_p is $Adv_{\mathscr{A}}^{ECDDHP}(t)$. Again, each message is linked with current timestamp, random nonce (short term secret), and long term secret. If $\mathscr{A}$ simulates the *Hash* queries $\mathscr{H}$ 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* quires and solving of ECDDHP. Applying birthday paradox to find the hash collision, the following result is obtained:

$$|Adv_{\mathscr{A},\mathbb{G}ame_1}^{AKMS-AgriIoT} - Adv_{\mathscr{A},\mathbb{G}ame_2}^{AKMS-AgriIoT}| \leq \frac{q_h^2}{2|\mathscr{H}|} + Adv_{\mathscr{A}}^{ECDDHP}(t_p). \tag{3}$$

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

$$Adv_{\mathscr{A},\mathbb{G}ame_2}^{AKMS-AgriIoT} = \frac{1}{2}. \tag{4}$$

Eq. (1) gives the semantic security of the proposed AKMS-AgriIoT as

$$\frac{1}{2}.Adv_{\mathscr{A}}^{AKMS-AgriIoT}(t_p) = \left|Adv_{\mathscr{A},\mathbb{G}ame_0}^{AKMS-AgriIoT} - \frac{1}{2}\right|. \tag{5}$$

Eqs. (2), (3) and (4), and use of triangular inequality lead to the following derivation from Eq. (5):

$$\begin{aligned} & \frac{1}{2}.Adv_{\mathscr{A}}^{AKMS-AgriIoT}(t_p) \\ = \; & |Adv_{\mathscr{A},\mathbb{G}ame_0}^{AKMS-AgriIoT} - Adv_{\mathscr{A},\mathbb{G}ame_2}^{AKMS-AgriIoT}| \\ = \; & |Adv_{\mathscr{A},\mathbb{G}ame_1}^{AKMS-AgriIoT} - Adv_{\mathscr{A},\mathbb{G}ame_2}^{AKMS-AgriIoT}| \\ \leq \; & \frac{q_h^2}{2|\mathscr{H}|} + Adv_{\mathscr{A}}^{ECDDHP}(t_p). \end{aligned} \tag{6}$$

Finally, if we multiply both sides of Eq. (6) by "a factor of 2", we arrive to the final result:

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

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

4.2. Informal security analysis

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

4.2.1. Replay attack

In AKMS-AgriIoT, the current timestamps and the random numbers are utilized in both authentication and key management phases. If an adversary $\mathscr{A}$ 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 old timestamp, it is easily detected by the respective recipient, and that message is simply discarded. AKMS-AgriIoT is then resilient against "replay attack".

4.2.2. Man-in-the-middle (MiTM) attack

In this attack, an adversary $\mathscr{A}$ 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 $\mathscr{A}$ 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 $\mathscr{A}$ creates a new random secret a'_{SD_j} and a fresh timestamp TS'_{SD_j}. Similarly, it is also infeasible task for $\mathscr{A}$ 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.

4.2.3. Impersonation attacks

Assume an adversary $\mathscr{A}$ 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, $\mathscr{A}$ 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 $\mathscr{A}$ 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.

4.2.4. 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 secret credential of them 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".

4.2.5. Physical IoT smart device and drone capture attacks

According to the discussed threat model in Section 1.2, an adversary $\mathscr{A}$ may physically capture some of the smart devices as well as drones. $\mathscr{A}$ 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" [7]. 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".

4.2.6. Ephemeral secret leakage (ESL) attack

In this attack, we apply the CK-adversary model as discussed in threat model (Section 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 $\mathscr{A}$ 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 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.

4.2.7. Block verification in blockchain

In AKMS-AgriIoT, assume that a verifier $\mathscr{V}$ wishes to validate a block $Block_m$ stored in the blockchain (as shown in Figure 6). In order to do this verification task, $\mathscr{V}$ 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, $\mathscr{V}$ rejects the $Block_m$. Otherwise, $\mathscr{V}$ 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 for validating a block. In addition, since a block contains the hash value of the previous block, it is quite impractical task for any adversary to tamper (modify/update) the information stored in the block.

5. Formal security verification using AVISPA: simulation study

The "Automated Validation of Internet Security Protocols and Applications (AVISPA)" [19] 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 the AVISPA implements only the Dolev-Yao (DY) threat model, it can detect both "replay and man-in-the-middle (MiTM)" attacks.

A designed protocol requires to implement using a language based on temporal logic, called "High Level Protocol Specification Language (HLPSL)" that is used to simulate a protocol and test their 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 hierarchy, with each role consisting of parameters, states and transitions between states. It also allows to create 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: a) "OFMC (On-The-Fly Model Checker)", b) "CL-AtSe (Constraint Logic based Attack Searcher)", c) "SATMC (SAT-based Model Checker)", and d) "TA4SP (Tree Automata based on Automatic Approximations for the Analysis of Security protocols)". The detailed discussions on AVISPA and its HLPSL implementation are readily available in [19].

In our implementation, we have considered the two cases:

- **Case 1:** authentication phase between drone and smart device (discussed in Section 3.3)
- **Case 2:** key management phase between drone and *GSS* (discussed in Section 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" [65]. The simulation results shown in Figs. 9 and 10 clearly demonstrate that AKMS-AgriIoT is secure against replay and MiTM attacks.

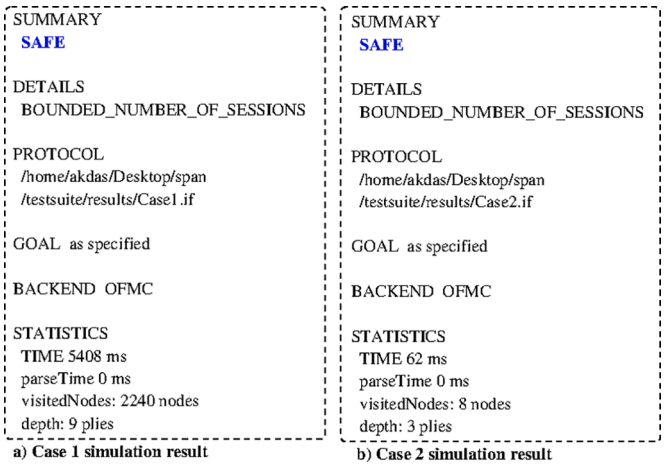

Fig. 9. Simulation results of AKMS-AgriIoT under OFMC backend for both cases (Case 1 and Case 2).

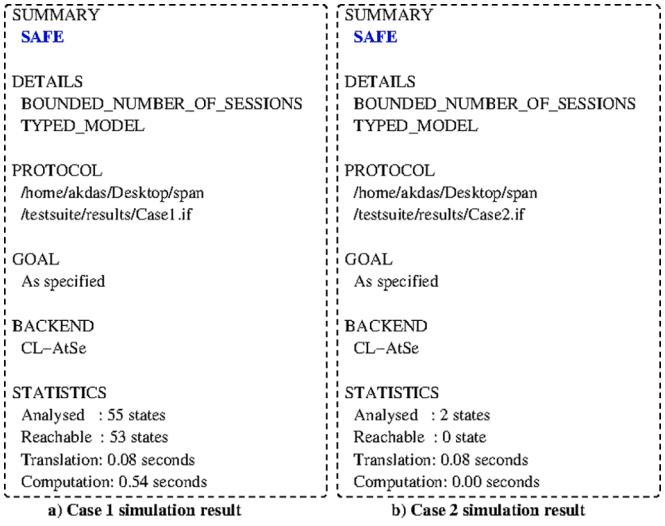

Fig. 10. Simulation results of AKMS-AgriIoT under CL-AtSe backend for both cases (Case 1 and Case 2).

6. Experimental results using MIRACL

Under this section, we evaluate various cryptographic primitives using the broadly-accepted "Multiprecision Integer and Rational Arithmetic Cryptographic Library (MIRACL)" [20] for their execution time. It is worth noticing that MIRACL, "a C/C++ based programming software library, has been already recognized by the cryptographers as the gold standard open source SDK for elliptic curve cryptography (ECC)".

We utilize the symbols T_{exp}, T_{ecm}, T_{eca}, T_{senc}/T_{sdec}, T_h, T_{mul} and T_{add} to signify the time required for "modular exponentiation", "elliptic curve point (scalar) multiplication", "elliptic curve point addition", "symmetric key (Advanced Encryption Standard (AES-128)) encryption/decryption", "one-way hash function using SHA-256 hashing algorithm", "modular multiplication over $GF(q)$" and "modular addition over $GF(q)$", respectively. The elliptic curve point addition and multiplication are carried out 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 4, we have then tabulated the experimental results.

- **Scenario 2:** In this case, we consider the platform for a smart device/drone as 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 [66]". Similar to Scenario 1, the experiments for each cryptographic primitive 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 5.

7. 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. 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. [39], Ali et al. [36], Tian et al. [34], Sadhukhan et al. [41], Shuai et al. [42], and Wu and Tsai [43].

7.1. 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 6. 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. [39], Ali et al. [36], Tian et al. [34], Sadhukhan et al. [41], Shuai et al. [42] and Wu and Tsai [43].

7.2. 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 Case 1 due to two messages $Msg_{SD_1} = \{TID_{SD_j}, A_{SD_j}, Sign_{SD_j}, TS_{SD_j}\}$, $Msg_{SD_2} = \{TID^*_{SD_j}, B_{DR_i}, Sign_{DR_{i_1}}, TS_{DR_{i_1}}\}$ 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 7 exhibits that AKMS-AgriIoT requires low communication costs as compared to other schemes for both the cases (Case 1 and Case 2).

Table 4
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 5
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

7.3. 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 [67], 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 Table 4 using MIRACL for the *GSS* or a server side. On the other side, we utilize the experiment results provided in Table 5 using MIRACL for a smart device or drone side. In both the 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 [68] has been applied for biometric verification. We denote T_{fe} as the time required for fuzzy extractor function. It is assumed that $T_{fe} \approx T_{ecm}$. The comparative study provided in Table 8 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 blockchain service. However, AKMS-AgriIoT requires significantly low communication costs and provides better security and functionality features provided in Table 6 as compared to all other compared schemes.

8. Blockchain implementation

This section gives the blockchain implementation of the proposed scheme (AKMS-AgriIoT). During the simulation, if the number of transactions reaches to a pre-defined transaction threshold (n_t), we select a leader L from the Peer-to-Peer (P2P) cloud servers (CS) network by 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 Algorithm 1, the leader L adds a block, say $Block_m$ as shown in Fig. 6, into the blockchain.

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 [69]. We have calculated a block size shown in Fig. 6 as follows: block version ($BVer_m$), previous block hash (PBH_m), Merkle tree root (MTR_m), timestamp

Table 6
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.

Table 7
Comparison of communication costs.

Protocol	No. of messages	Total cost (in bits)
Tai et al. [39]	4	2560
Ali et al. [36]	5	5504
Tian et al. [34]	2	$384s + 11712$
Sadhukhan et al. [41]	4	5248
Shuai et al. [42]	4	7616
Wu and Tsai [43]	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" [34]; *n*: "number of agricultural equipment (sensor devices) and blockchains in Wu and Tsai's scheme" [43]

Table 8
Comparison of computation costs.

Protocol	Smart device/Drone end	GSS/Server end
Ali et al. [36]	$11T_h + T_{fe} + 3T_{senc}/T_{sdec}$	$8T_h + 5T_{senc}/T_{sdec}$
	≈ 5.735 ms	≈ 0.445 ms
Tian et al. [34]	$8T_{exp} + 9T_h$	-
	≈ 4.605 ms	
Tai et al. [39]	$17T_h$	$6T_h$
	≈ 5.253 ms	≈ 0.330 ms
Sadhukhan et al. [41]	$4T_h + 2T_{enc} + 2T_{dec} + 2T_{ecm}$	$2T_h + 2T_{dec} + 2T_{enc}$
	≈ 5.876 ms	≈ 0.114 ms
Shuai et al. [42]	$13T_h + 3T_{exp}$	$7T_h + T_{exp}$
	≈ 4.701 ms	≈ 0.457 ms
Wu and Tsai [43]	$2T_{bp} + 2T_{exp} + 2T_{enc/dec} + T_h$	$2T_{bp} + 2T_{exp} + 2T_{enc/dec} + T_h$
	≈ 64.965 ms	≈ 9.407 ms
AKMS-AgriIoT (Case 1)	$11T_h + 8T_{ecm} + 2T_{eca}$	-
	≈ 21.735 ms	
AKMS-AgriIoT (Case 2)	$7T_h + 6T_{ecm} + 2T_{eca} + T_{poly}$	$7T_h + 6T_{ecm} + 2T_{eca} + T_{poly}$
	≈ 18.023 ms	≈ 4.733 ms

(epoch time) (TS_m), owner of the block (OB_m), public key of signer (Pub_{GSS}), encrypted transaction ($E_{Pub_{GSS}}(Tx_i)$) by applying the ECC encryption, 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, 42 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 9, we have listed the performance measures relevant to the voting-based PBFT consensus algorithm. Note that Algorithm 1 requires 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 Algorithm 1 is $4n^2 = O(n^2)$, where n is the total number of P2P nodes.

The details simulation are provided in two scenarios as shown In Figs. 11 and 12, we have shown the blockchain simulation results for the proposed AKMS-AgriIoT. We have considered the following two scenarios, where we have taken the total number of peer nodes in the P2P network as 13.

- **Scenario 1:** This scenario considers the number of transactions per block as 25. The simulation results provided in Fig. 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 Fig. 12 illustrates that the total computational time increases linearly when the number of transactions per block increases during the consensus process.

Table 9
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	$6T_h + 12T_{ecm} + 6T_{eca}$

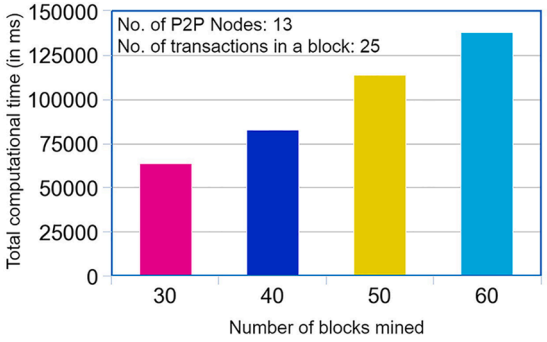

Fig. 11. Blockchain simulation results for Scenario 1.

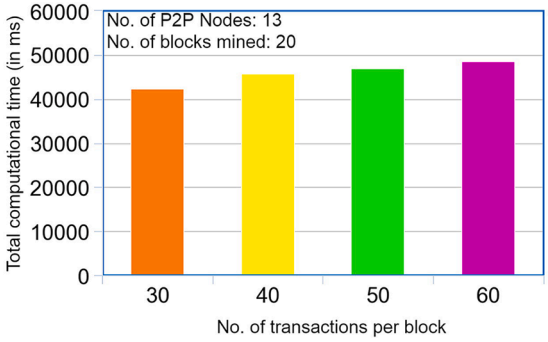

Fig. 12. Blockchain simulation results for Scenario 2.

- **Scenario 3:** In this scenario, we consider the number of blocks is 25 and the number of transactions is 30 under different P2P nodes in the virtual distributed system. The simulation results are shown in Fig. 13. When the number of P2P nodes is varied, the total computational time (in milliseconds) also increase linearly.

9. Conclusion

This work is an attempt to propose an interesting security scheme, namely the blockchain-based authentication and key agreement in 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 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 in 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 among "security and functionality features", and "communication and computation overheads" as compared to other schemes.

CRediT authorship contribution statement

Basudeb Bera: Conceptualization, Methodology, Software, Validation, Formal analysis, Writing – original draft. **Anusha Vangala:** Conceptualization, Methodology, Software, Validation, Writing – original draft. **Ashok Kumar Das:** Supervision, Visualization, Conceptualization, Investigation, Writing – original draft, Writing – review & editing. **Pascal Lorenz:** Software, Validation, Visualization, Investigation. **Muhammad Khurram Khan:** Visualization, Investigation.

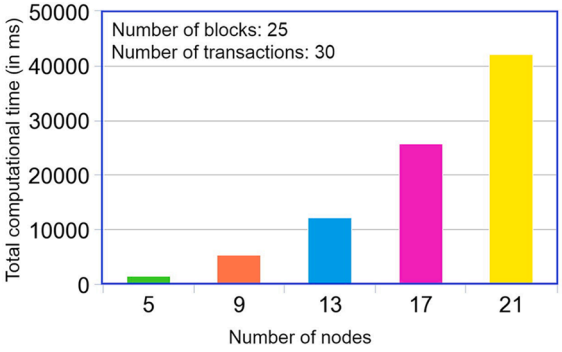

Fig. 13. Blockchain simulation results for Scenario 3.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors thank the anonymous reviewers, the associate editor and the editor-in-chief for their valuable feedback on the paper, which helped us to improve its quality and presentation. This work was supported by the Mathematical Research Impact Centric Support (MATRICS) project funded by the Science and Engineering Research Board (SERB), India (Reference No. MTR/2019/000699) and also by the Ripple Centre of Excellence Scheme, CoE in Blockchain (Sanction No. IIIT/R&D Office/Internal Projects/001/2019), IIIT Hyderabad, India. The authors are also supported by Researchers Supporting Project number (RSP-2021/12), King Saud University, Riyadh, Saudi Arabia.

References

[1] U. Shafi, R. Mumtaz, J. Garcia-Nieto, S.A. Hassan, S.A.R. Zaidi, N. Iqbal, Precision agriculture techniques and practices: from considerations to applications, Sensors 19 (17) (2019).
[2] A. Khanna, S. Kaur, Evolution of internet of things (IoT) and its significant impact in the field of precision agriculture, Comput. Electron. Agric. 157 (2019) 218–231.
[3] X. Shi, X. An, Q. Zhao, H. Liu, L. Xia, X. Sun, et al., State-of-the-art internet of things in protected agriculture, Sensors 19 (8) (2019).
[4] V. Saiz-Rubio, F. Rovira-Mas, From smart farming towards agriculture 5.0: a review on crop data management, Agronomy 10 (2) (2020).
[5] D. Dolev, A. Yao, On the security of public key protocols, IEEE Trans. Inf. Theory 29 (2) (1983) 198–208.
[6] R. Canetti, H. Krawczyk, Universally composable notions of key exchange and secure channels. International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT'02), Amsterdam, The Netherlands, 2002, pp. 337–351.
[7] T.S. Messerges, E.A. Dabbish, R.H. Sloan, Examining smart-card security under the threat of power analysis attacks, IEEE Trans. Comput. 51 (5) (2002) 541–552.
[8] L. Ge, C. Brewster, J. Spek, A. Smeenk, J. Top, F. van Diepen, et al., Blockchain for Agriculture and Food: Findings from the Pilot Study. 112, Wageningen Economic Research, 2017.
[9] M. Torky, A.E. Hassanein, Integrating blockchain and the internet of things in precision agriculture: analysis, opportunities, and challenges, Comput. Electron. Agric. 178 (2020) 105476.
[10] S.V. Akram, P.K. Malik, R. Singh, G. Anita, S. Tanwar, Adoption of blockchain technology in various realms: opportunities and challenges, Secur. Privacy 3 (5) (2020) e109, https://doi.org/10.1002/spy2.109.
[11] Y.P. Lin, J.R. Petway, J. Anthony, H. Mukhtar, S.W. Liao, C.F. Chou, et al., Blockchain: the evolutionary next step for ICT e-agriculture, Environments 4 (3) (2017), https://doi.org/10.3390/environments4030050.
[12] P. Tripicchio, M. Satler, G. Dabisias, E. Ruffaldi, C.A. Avizzano, Towards smart farming and sustainable agriculture with drones. International Conference on Intelligent Environments, Prague, Czech Republic, 2015, pp. 140–143.
[13] N.J. Stehr, Drones: the newest technology for precision agriculture, Nat. Sci. Educ. 44 (1) (2015) 89–91.
[14] V. Puri, A. Nayyar, L. Raja, Agriculture drones: a modern breakthrough in precision agriculture, J. Stat. Manage. Syst. 20 (4) (2017) 507–518.
[15] M. Kulbacki, J. Segen, W. Knie, R. Klempous, K. Kluwak, J. Nikodem, et al., Survey of drones for agriculture automation from planting to harvest. 22nd International

Conference on Intelligent Engineering Systems (INES), Las Palmas de Gran Canaria, Spain, 2018, pp. 353–358, https://doi.org/10.1109/INES.2018.8523943.
[16] M. Ayamga, B. Tekinerdogan, A. Kassahun, Exploring the challenges posed by regulations for the use of drones in agriculture in the african context, Land 10 (2) (2021), https://doi.org/10.3390/land10020164.
[17] D. van der Merwe, D.R. Burchfield, T.D. Witt, K.P. Price, A. Sharda, Chapter one - drones in agriculture. vol. 162 of Advances in Agronomy, Academic Press, 2020, pp. 1–30, https://doi.org/10.1016/bs.agron.2020.03.001.
[18] e-agriculture in action: drones for agriculture, 2018, http://www.fao.org/3/i8494en/i8494en.pdf.
[19] AVISPA, Automated validation of internet security protocols and applications, 2019, Http://www.avispa-project.org/. Accessed on October 2019.
[20] MIRACL cryptographic SDK: Multiprecision integer and rational arithmetic cryptographic library, 2020, https://github.com/miracl/MIRACL, Accessed on April 2020.
[21] A.K. Das, P. Sharma, S. Chatterjee, J.K. Sing, A dynamic password-based user authentication scheme for hierarchical wireless sensor networks, J. Netw. Comput. Appl. 35 (5) (2012) 1646–1656.
[22] S. Chatterjee, A.K. Das, J.K. Sing, An enhanced access control scheme in wireless sensor networks, Ad Hoc Sens. Wirel. Netw. 21 (1-2) (2014) 121–149.
[23] V. Odelu, A.K. Das, A. Goswami, SEAP: secure and efficient authentication protocol for NFC applications using pseudonyms, IEEE Trans. Consum. Electron. 62 (1) (2016) 30–38.
[24] D. Mishra, A.K. Das, S. Mukhopadhyay, A secure and efficient ECC-based user anonymity-preserving session initiation authentication protocol using smart card, Peer-to-Peer Netw. Appl. 9 (1) (2016) 171–192.
[25] M. Wazid, A.K. Das, M.K. Khan, A.A. Al-Ghaiheb, N. Kumar, A.V. Vasilakos, Secure authentication scheme for medicine anti-counterfeiting system in IoT environment, IEEE Internet Things J. 4 (5) (2017) 1634–1646.
[26] A.K. Das, S. Kumari, V. Odelu, X. Li, F. Wu, X. Huang, Provably secure user authentication and key agreement scheme for wireless sensor networks, Secur. Commun. Netw. 9 (16) (2016) 3670–3687.
[27] F. Wu, L. Xu, S. Kumari, X. Li, J. Shen, K.K.R. Choo, et al., An efficient authentication and key agreement scheme for multi-gateway wireless sensor networks in IoT deployment, J. Netw. Comput. Appl. 89 (2017) 72–85.
[28] M. Wazid, A.K. Das, V. Bhat K, A.V. Vasilakos, LAM-CIoT: Lightweight authentication mechanism in cloud-based IoT environment, J. Netw. Comput. Appl. 150 (2020) 102496.
[29] M. Wazid, A.K. Das, N. Kumar, A.V. Vasilakos, J.J.P.C. Rodrigues, Design and analysis of secure lightweight remote user authentication and key agreement scheme in internet of drones deployment, IEEE Internet Things J. 6 (2) (2019) 3572–3584.
[30] S. Roy, S. Chatterjee, A.K. Das, S. Chattopadhyay, N. Kumar, A.V. Vasilakos, On the design of provably secure lightweight remote user authentication scheme for mobile cloud computing services, IEEE Access 5 (2017) 25808–25825.
[31] S. Challa, A.K. Das, P. Gope, N. Kumar, F. Wu, A.V. Vasilakos, Design and analysis of authenticated key agreement scheme in cloud-assisted cyber-physical systems, Future Gener. Comput. Syst. 108 (2020) 1267–1286.
[32] M. Wazid, A.K. Das, A.V. Vasilakos, Authenticated key management protocol for cloud-assisted body area sensor networks, J. Netw. Comput. Appl. 123 (2018) 112–126.
[33] D. Wang, W. Li, P. Wang, Measuring two-factor authentication schemes for real-time data access in industrial wireless sensor networks, IEEE Trans. Ind. Inf. 14 (9) (2018) 4081–4092.
[34] Y. Tian, J. Yuan, H. Song, Efficient privacy-preserving authentication framework for edge-assisted internet of drones, J. Inf. Secur. Appl. 48 (2019) 102354.
[35] M. Wazid, P. Bagga, A.K. Das, S. Shetty, J.J.P.C. Rodrigues, Y. Park, AKM-IoV: authenticated key management protocol in fog computing-based internet of vehicles deployment, IEEE Internet Things J. 6 (5) (2019) 8804–8817.
[36] R. Ali, A.K. Pal, S. Kumari, M. Karuppiah, M. Conti, A secure user authentication and key-agreement scheme using wireless sensor networks for agriculture monitoring, Future Gener. Comput. Syst. 84 (2018) 200–215.
[37] F. Wu, L. Xu, S. Kumari, X. Li, A new and secure authentication scheme for wireless sensor networks with formal proof, Peer-to-Peer Netw. Appl. 10 (2017) 16–30.
[38] J. Srinivas, S. Mukhopadhyay, D. Mishra, Secure and efficient user authentication scheme for multi-gateway wireless sensor networks, Ad Hoc Netw. 54 (2017) 147–169.
[39] W.L. Tai, Y.F. Chang, W.H. Li, An IoT notion-based authentication and key agreement scheme ensuring user anonymity for heterogeneous ad hoc wireless sensor networks, J. Inf. Secur. Appl. 34 (2017) 133–141.
[40] R. Almadhoun, M. Kadadha, M. Alhemeiri, M. Alshehhi, K. Salah, A user authentication scheme of IoT devices using blockchain-enabled fog nodes. IEEE/ACS 15th International Conference on Computer Systems and Applications (AICCSA), Aqaba, Jordan, 2018, pp. 1–8.
[41] D. Sadhukhan, S. Ray, G. Biswas, M. Khan, M. Dasgupta, A lightweight remote user authentication scheme for IoT communication using elliptic curve cryptography, J. Supercomput. (2020), https://doi.org/10.1007/s11227-020-03318-7.
[42] M. Shuai, L. Xiong, C. Wang, N. Yu, A secure authentication scheme with forward secrecy for industrial internet of things using Rabin cryptosystem, Comput. Commun. 160 (2020) 215–227.
[43] H. Wu, C. Tsai, An intelligent agriculture network security system based on private blockchains, J. Commun. Netw. 21 (5) (2019) 503–508.
[44] N. Drljevic, D.A. Aranda, V. Stantchev, Perspectives on risks and standards that affect the requirements engineering of blockchain technology, Comput. Standards Interfaces 69 (2020) 103409.
[45] L. Tan, H. Xiao, K. Yu, M. Aloqaily, Y. Jararweh, A blockchain-empowered crowdsourcing system for 5G-enabled smart cities, Comput. Standards Interfaces 76 (2021) 103517.
[46] C. Zhang, C. Xu, K. Sharif, L. Zhu, Privacy-preserving contact tracing in 5G-integrated and blockchain-based medical applications, Comput. Standards Interfaces 77 (2021) 103520.
[47] X. Ma, C. Wang, X. Chen, Trusted data sharing with flexible access control based on blockchain, Comput. Standards Interfaces 78 (2021) 103543.
[48] D. Liu, Y. Zhang, D. Jia, Q. Zhang, X. Zhao, H. Rong, Toward secure distributed data storage with error locating in blockchain enabled edge computing, Comput. Standards Interfaces 79 (2022) 103560.
[49] Q. Jiang, S. Zeadally, J. Ma, D. He, Lightweight three-factor authentication and key agreement protocol for internet-integrated wireless sensor networks, IEEE Access 5 (2017) 3376–3392.
[50] L. Wu, J. Wang, S. Zeadally, D. He, Anonymous and efficient message authentication scheme for smart grid, Secur. Commun. Netw. 2019 (2019) 4836016:1–4836016:12.
[51] M. Castro, B. Liskov, Practical byzantine fault tolerance and proactive recovery, ACM Trans. Comput. Syst. 20 (4) (2002) 398–461.
[52] W.E. May, Secure hash standard. FIPS PUB 180-1, National Institute of Standards and Technology (NIST), U.S. Department of Commerce, 2015.http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf
[53] D. Johnson, A. Menezes, S. Vanstone, The elliptic curve digital signature algorithm (ECDSA), Int. J. Inf. Secur. 1 (1) (2001) 36–63.
[54] C. Blundo, A. De Santis, A. Herzberg, S. Kutten, U. Vaccaro, M. Yung, Perfectly secure key distribution for dynamic conferences, Inf. Comput. 146 (1) (1998) 1–23.
[55] A.K. Das, An unconditionally secure key management scheme for large-scale heterogeneous wireless sensor networks. First International Communication Systems and Networks and Workshops (COMSNETS'09), Bangalore, India, 2009, pp. 1–10.
[56] H. Zhang, J. Wang, Y. Ding, Blockchain-based decentralized and secure keyless signature scheme for smart grid, Energy 180 (2019) 955–967.
[57] D. Wang, D. He, P. Wang, C. Chu, Anonymous two-factor authentication in distributed systems: certain goals are beyond attainment, IEEE Trans. Dependable Secure Comput. 12 (4) (2015) 428–442.
[58] M. Burrows, M. Abadi, R. Needham, A logic of authentication, ACM Trans. Comput. Syst. 8 (1) (1990) 18–36.
[59] M. Abdalla, P.A. Fouque, D. Pointcheval, Password-based authenticated key exchange in the three-party setting. 8th International Workshop on Theory and Practice in Public Key Cryptography (PKC'05), Lecture Notes in Computer Science vol. 3386. Les Diablerets, Switzerland, 2005, pp. 65–84.
[60] C.C. Chang, H.D. Le, A provably secure, efficient, and flexible authentication scheme for ad hoc wireless sensor networks, IEEE Trans. Wirel. Commun. 15 (1) (2016) 357–366.
[61] M. Wazid, A.K. Das, S. Kumari, X. Li, F. Wu, Design of an efficient and provably secure anonymity preserving three-factor user authentication and key agreement scheme for TMIS, Secur. Commun. Netw. 9 (13) (2016) 1983–2001.
[62] J. Srinivas, A.K. Das, N. Kumar, J. Rodrigues, Cloud centric authentication for wearable healthcare monitoring system, IEEE Trans. Dependable Secure Comput. 17 (5) (2020) 942–956.
[63] M. Wazid, A.K. Das, N. Kumar, V. Odelu, A. Goutham Reddy, K. Park, et al., Design of lightweight authentication and key agreement protocol for vehicular ad hoc networks, IEEE Access 5 (2017) 14966–14980.
[64] J. Srinivas, A.K. Das, N. Kumar, J.J.P.C. Rodrigues, TCALAS: temporal credential-based anonymous lightweight authentication scheme for internet of drones environment, IEEE Trans. Veh. Technol. 68 (7) (2019) 6903–6916.
[65] AVISPA, SPAN, the security protocol ANimator for AVISPA, 2019, Http://www.avispa-project.org/. Accessed on October 2019.
[66] Raspberry Pi 3 Model B+, 2020, https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/, Accessed on April 2020.
[67] D.E. Knuth, The Art of Computer Programming: Seminumerical Algorithms vol. 2, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1997.
[68] Y. Dodis, L. Reyzin, A. Smith, Fuzzy extractors: how to generate strong keys from biometrics and other noisy data. International Conference on the Theory and Applications of Cryptographic Techniques (Eurocrypt'04), Interlaken, Switzerland, 2004, pp. 523–540.
[69] K. Khullar, Implementing PBFT in blockchain, 2019, https://medium.com/coinmonks/implementing-pbft-in-blockchain-12368c6c9548, Accessed on August 2020.

Basudeb Bera received his M.Sc. degree in mathematics and computing in 2014 from IIT (ISM) Dhanbad, India, and M.Tech. degree in computer science and data processing in 2017 from IIT Kharagpur, India. He is currently pursuing his Ph.D. degree in computer science and engineering from the Center for Security, Theory and Algorithmic Research, IIIT Hyderabad, India. His research interests are cryptography, network security and blockchain technology. He has published more than 15 papers in international journals and conferences in his research areas.

Anusha Vangala received her M.Tech. degree in computer science and engineering from Jawaharlal Nehru Technological University, Kakinada, India. She is currently pursuing her Ph.D. degree in computer science and engineering from the Center for Security, Theory and Algorithmic Research, IIIT Hyderabad, India. Prior to joining at IIIT Hyderabad for Ph.D. program, she had nearly 5 years of teaching experience as an assistant professor in computer science and engineering at various renowned institutes across India. Her research interests include cloud computing, Internet of Things (IoT) and blockchain technology. She has published more than 10 papers in international journals and conferences in her research areas.

Ashok Kumar Das received a Ph.D. degree in computer science and engineering, an M.Tech. degree in computer science and data processing, and an M.Sc. degree in mathematics from IIT Kharagpur, India. He is currently an Associate Professor with the Center for Security, Theory and Algorithmic Research, International Institute of Information Technology, Hyderabad, India. His current research interests include cryptography, network security, security in vehicular ad hoc networks, smart grids, smart homes, Internet of Things (IoT),Internet of Drones, Internet of Vehicles, Cyber-Physical Systems (CPS) and cloud computing, intrusion detection, blockchain and AI/ML security. He has authored over 270 papers in international journals and conferences in the above areas, including over 230 reputed journal papers. Some of his research findings are published in top cited journals, such as the IEEE Transactions on Information Forensics and Security, IEEE Transactions on Dependable and Secure Computing, IEEE Transactions on Smart Grid, IEEE Transactions on Intelligent Transportation Systems, IEEE Internet of Things Journal, IEEE Transactions on Industrial Informatics, IEEE Transactions on Vehicular Technology, IEEE Transactions on Consumer Electronics, IEEE Journal of Biomedical and Health Informatics (formerly IEEE Transactions on Information Technology in Biomedicine), IEEE Consumer Electronics Magazine, IEEE Access, IEEE Communications Magazine, Future Generation Computer Systems, Computers & Electrical Engineering, Computer Methods and Programs in Biomedicine, Computer Standards & Interfaces, Computer Networks, Expert Systems with Applications, and Journal of Network and Computer Applications. He was a recipient of the Institute Silver Medal from IIT Kharagpur. He is on the editorial board of IEEE Systems Journal, Journal of Network and Computer Applications (Elsevier), Computer Communications (Elsevier), IET Communications, KSII Transactions on Internet and Information Systems, and International Journal of Internet Technology and Secured Transactions (Inderscience), is a Guest Editor for Computers & Electrical Engineering (Elsevier) for the special issue on Big data and IoT in e-healthcare and for ICT Express (Elsevier) for the special issue on Blockchain Technologies and Applications for 5G Enabled IoT, and has served as a Program Committee Member in many international conferences.He also severed as one of the Technical Program Committee Chairs of the first International Congress on Blockchain and Applications (BLOCKCHAIN'19), Avila, Spain, June 2019, International Conference on Applied Soft Computing and Communication Networks (ACN'20), October 2020, Chennai, India, and second International Congress on Blockchain and Applications (BLOCKCHAIN'20), L'Aquila, Italy, October 2020. He is a senior member of the IEEE.

Pascal Lorenz received his M.Sc. (1990) and Ph.D. (1994) from the University of Nancy, France. Between 1990 and 1995 he was a research engineer at WorldFIP Europe and at Alcatel-Alsthom. He is a professor at the University of Haute-Alsace, France, since 1995. His research interests include QoS, wireless networks and high-speed networks. He is the author/co-author of 3 books, 3 patents and 200 international publications in refereed journals and conferences. He was Technical Editor of the IEEE Communications Magazine Editorial Board (2000-2006), IEEE Networks Magazine since 2015, IEEE Transactions on Vehicular Technology since 2017, Chair of IEEE ComSoc France (2014-2018), Financial chair of IEEE France (2017-2019), Chair of Vertical Issues in Communication Systems Technical Committee Cluster (2008-2009), Chair of the Communications Systems Integration and Modeling Technical Committee (2003-2009), Chair of the Communications Software Technical Committee (2008-2010) and Chair of the Technical Committee on Information Infrastructure and Networking (2016-2017). He is associate Editor for International Journal of Communication Systems (IJCS-Wiley), Journal on Security and Communication Networks (SCN-Wiley) and International Journal of Business Data Communications and Networking, Journal of Network and Computer Applications (JNCA-Elsevier). He is senior member of the IEEE, IARIA fellow and member of many international program committees.

Muhammad Khurram Khan is currently working as a Professor of Cybersecurity at the Center of Excellence in Information Assurance, King Saud University, Kingdom of Saudi Arabia. He is founder and CEO of the 'Global Foundation for Cyber Studies and Research' (http://www.gfcyber.org), an independent and non-partisan cybersecurity think-tank in Washington D.C, USA. He is the Editor-in-Chief of 'Telecommunication Systems' published by Springer-Nature with its recent impact factor of 1.73 (JCR 2020). He is on the editorial board of several journals including, IEEE Communications Surveys & Tutorials, IEEE Communications Magazine, IEEE Internet of Things Journal, IEEE Transactions on Consumer Electronics, Journal of Network & Computer Applications (Elsevier), IEEE Access, IEEE Consumer Electronics Magazine, PLOS ONE, and Electronic Commerce Research, etc. He has published more than 380 papers in the journals and conferences of international repute. In addition, he is an inventor of 10 US/PCT patents. He has edited 10 books/proceedings published by Springer-Verlag, Taylor & Francis and IEEE. His research areas of interest are Cybersecurity, digital authentication, IoT security, biometrics, multimedia security, cloud computing security, cyber policy, and technological innovation management. He is a fellow of the IET (UK), a fellow of the BCS (UK), and a fellow of the FTRA (Korea). He is the Vice Chair of IEEE Communications Society Saudi Chapter. He is a distinguished Lecturer of the IEEE. His detailed profile can be visited at http://www.professorkhurram.com.