Public Blockchain-Envisioned Security Scheme Using Post Quantum Lattice-Based Aggregate Signature for Internet of Drones Applications

Prithwi Bagchi, Raj Maheshwari, Basudeb Bera, Ashok Kumar Das, *Senior Member, IEEE*, Youngho Park, *Member, IEEE*, Pascal Lorenz, *Senior Member, IEEE*, and David K. Y. Yau, *Senior Member, IEEE*

***Abstract*—Due to high effectiveness and robust security protocols, lattice-based cryptography becomes a very broadly applicable optimistic post-quantum technique that is recently used in public key cryptosystem. An aggregate signature scheme enables a party to bundle a set of signatures together into a single short cryptographic signature, which can be verified by any verifier using the public information. In this paper, we provide a lattice-based aggregate signature scheme where the security depends on the difficulty of the Ring Learning-with-Error (Ring-LWE) problem. Next, we use the basic scheme in Internet of Drones (IoD) applications using the blockchain technology for secure and transparent data storage. The detailed security analysis and comparative study show that the proposed scheme provides superior security including resistance to quantum attacks and is efficient as compared to the existing state of art approaches. The testbed experimental results and the blockchain simulation demonstrate that the proposed scheme can be applied in real-life drones applications.**

***Index Terms*—Aggregate signature, blockchain, Internet of Drones (IoD), lattice-based cryptography, security, unmanned aerial vehicles.**

I. Introduction

THE exposure of quantum computers becomes a menacing on the security of the traditional public-key cryptosystem. Shor [1] mentioned that the existing mathematical number theoretic hard problems, like "Integer Factorization Problem (IFP) in RSA public key cryptosystem" and "Discrete Logarithm Problem (DLP) in ElGamal cryptosystem" are not safe under the quantum computing scenario. Traditional digital signature is a basic cryptographic primitive which has various applications, like verifying the authenticity and integrity of the received messages. One of most noteworthy post-quantum cryptosystem that has broadly applicable on traditional security domain is the "Lattice-based Cryptography (LBC)". The security of the quantum resistance lattice-based signature scheme is based on the difficulty of some problems, like "Shortest Integer Solution (SIS)" assumption and "Learning With Error (LWE)" assumption [2]. The SIS and LWE assumptions are both computational assumptions. Generally, the LWE-oriented lattice-based signature schemes contain an enormous memory. To reduce the memory space, a new trend has been introduced, which is called the ring lattice-based signature scheme (Ring-LBS), where the security of a Ring-LBS scheme is based on the Ring-LWE and Ring-SIS assumptions.

An aggregate signature [3] basically merges a group of signatures which are connected with different messages and forms a single compact signature. The aggregate signature provides authenticity to the group of message-signature pairs corresponding to the group of signers in a single step. It is primarily used to reduce the required storage space of the signature, and hence, it reduces the required network bandwidth for transmission and the number of signature verification computations. Thus, for designing a quantum resistance aggregate signature scheme, a lattice-based aggregate signature scheme plays a significant role [4].

The Internet of Things (IoT) is considered as a system of interrelated computing smart devices or objects (things) that are provided with "unique identifiers (UIDs)" and the connected devices have the ability to send the data over a network without needing "human-to-human" or "human-to-computer interaction". The drones, also called as unmanned aerial vehicles (UAV), related futuristic disruptive technologies, such as "quantum drones (QD)," "Internet of Quantum Drones (IoQDs)," and "constellation of quantum satellites (CQS)" are supposed to be a breakthrough technology in strategic fields of society [5]. It is also expected that the "quantum device-integrated over drones" can be developed in near future for any storage, computation and communication [5].

Manuscript received 25 August 2022; revised 20 December 2022 and 19 February 2023; accepted 20 March 2023. Date of publication 22 March 2023; date of current version 15 August 2023. This work was supported in part by the National Research Foundation of Korea through Basic Science Research Program, Ministry of Education under Grant 2020R1I1A3058605 and in part by the Blockchain Project under the Ripple Centre of Excellence (CoE) Scheme, CoE in Blockchain, IIIT Hyderabad, India under Grant IIIT/R&D Office/Internal Projects/001/2019. The review of this article was coordinated by Prof. Jalel Ben-Othman. *(Corresponding authors: Youngho Park; Pascal Lorenz.)*

Prithwi Bagchi, Raj Maheshwari, and Ashok Kumar Das are with the Center for Security, Theory and Algorithmic Research, International Institute of Information Technology, Hyderabad 500032, India (e-mail: prithwi.bagchi@research.iiit.ac.in; raj.maheshwari@students.iiit.ac.in; iitkgp.akdas@gmail.com).

Basudeb Bera is with the Information Systems Technology and Design, Singapore University of Technology and Design, Singapore 487372 (e-mail: basudeb_bera@sutd.edu.sg).

Youngho Park is with the School of Electronics Engineering, Kyungpook National University, Daegu 41566, South Korea (e-mail: parkyh@knu.ac.kr).

Pascal Lorenz is with the University of Haute Alsace, 68000 Colmar, France (e-mail: lorenz@ieee.org).

David K. Y. Yau is with the Information Systems Technology and Design, Singapore University of Technology and Design, Singapore 487372 (e-mail: david_yau@sutd.edu.sg).

Digital Object Identifier 10.1109/TVT.2023.3260579

In an Internet of Drones (IoD) environment, multiple drones are deployed in a particular flying zone or an application area and they are connected in the IoD environment [6], [7], [8]. As a result, an aggregate signature scheme is considered as one of promising security solutions for resolving privacy and security issues in the IoD network [9]. The drones in an application collect the data and send their data along with the individual signatures to an aggregator node, called the ground station server (GSS). The GSS then validates all the individual signatures and forms an aggregate signature on the collected signed messages. The aggregate message along with their compact aggregate signature are then sent to the cloud server for data storage purpose into the public blockchain. Any verifier can verify the data integrity of the aggregate message by means of applying the aggregate signature verification algorithm.

A. Research Contributions

In the following, we list the main contributions made in this article:

- An efficient lattice-based aggregate signature scheme has been proposed, whose security is based on the hardness of the lattice-based "Ring-Learning-with-Error (LWE) problem". The designed lattice-based aggregate signature scheme is then applied for the real-time IoD applications, where the drones play the role of individual signers and the GSS plays the role of an aggregator. After verifying the signatures, the in-charge cloud server in the "Peer-to-Peer (P2P) cloud servers network" creates blocks and mine them using the voting-based consensus algorithm to add in the public blockchain for secure data storage.
- The formal and informal security analysis has been performed on the proposed scheme to show its robustness against various attacks including the quantum attacks based on the proposed threat model discussed in Section II-B.
- A testbed experiment has been conducted to implement the proposed lattice-based aggregated signature scheme (LBAS) to measure computational time needed for single and aggregate signature generation and verification under the setting of a drone, a GSS and a cloud server.
- A blockchain simulation has been provided to measure the computational time needed for mining the blocks in the blockchain by varying the number of transactions and blocks in the chain. A comprative study also reveals the efficiency and security of the proposed scheme over the existing competing schemes.

B. Outline

The various systems models like network and threat models are presented in Section II. Using the mathematical preliminaries presented in Section III, we present the lattice-based aggregate signature scheme (LBAS) in Section IV. Next, in Section V, we describe how the designed LBAS is applied for the real-time IoT-enabled drones applications using the public blockchain. The formal and informal security analysis of the proposed scheme is presented in Section VI. Sections VII and VIII provide the real testbed experiments and blockchain simulation of the proposed scheme, respectively. After that, the related work and a comparative analysis are conducted in Section IX. Some concluding remarks are finally drawn in Section X.

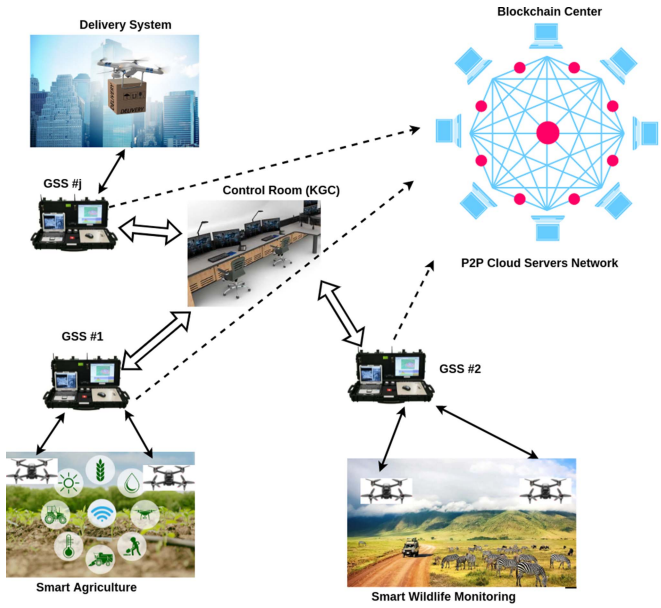


Fig. 1. Network model used in IoD applications.

II. System Models

In this section, we first mention the network model used in the proposed blockchain-based aggregate signature scheme for IoD applications. Next, we discuss the threat model that is applied for analyzing the security of the proposed scheme.

A. Network Model

In the proposed scheme, we consider a network model as shown in Fig. 1. In the network model, the participating entities in the network include: 1) drones, 2) ground station servers (GSS), 3) control room, acting as the key generation center (KGC), and 4) cloud servers. IoD has several potential applications as mentioned in Fig. 1. For example, consider the delivery system using drones. In such a scenario, drones play an important role as "a valid alternative to support the delivery method and various big companies, like Amazon and DHL, have initiated to apply the drones for their parcels deliveries" [10]. Due to difficult and inaccessible terrains, there are some areas which may not be easily accessible for collection of data. In this case, the drones can be deployed in those areas to efficiently aggregate more detailed information with less time and low cost [11]. The use of drones in ecology also helps in smart wildlife monitoring to see the "impact of ongoing global change," improve the "Earth's biodiversity system," and also predict the "future trends of ecology and its development" [12]. Consider other important applications using the drones, such as "smart farming" and "Precision Agriculture (PA)," where the "aerial remote sensing" is typically most applied technology. Drone monitoring systems in such cases will be helpful to the farmers to take various observation with the help of the aerial views of the harvest. This

allows to gather various information, like the water system, soil variety, fungal infestations and pests [13].

The role of a drone is to gather the data from its deployed application area or flying zone and sign the message (sensing data) being a signer. The drones are equipped with the IoT-enabled smart devices like "Global Positioning System (GPS)," cameras, frames, flight controller and transceiver. The GSS is treated as an aggregator node whose task is to aggregate the individual signed messages from its member drones. The GSS performs individual signature verification, creates an aggregate signature on the received signed messages and then sends to a cloud server to store in the blockchain network consisting of the "Peer-to-Peer (P2P) cloud servers network" through the consensus algorithm. The in-charge cloud server checks the validity the aggregate signature, creates blocks and mine them for adding into the public blockchain.

The control room recognized as KGC has the responsibility to register all drones and GSSs by generating their private and public keys pairs using the lattice-based key generation algorithm. The control room is treated as a fully-trusted entity, whereas the GSSs are considered as semi-trusted entities in the network. Both the control room and the GSSs are placed under physical locking system in order to protect them from the attackers. The cloud servers in the P2P network are treated as semi-trusted entities. Once the registration process is over, the drones and the in-charge GSS can be deployed in their functioning zone. The cloud servers participated in the distributed system help in building a distributed technology for utilizing the public blockchain storage.

B. Threat Model

In this work, we consider the following three types of adversary models:

- *Honest-but-Curious adversary model:* Such as model [14] represents a passive adversarial model. In this model, an adversary, say $\mathcal{F}$ behaves as an authorized entity and follows a specified protocol. However, $\mathcal{F}$ can still intercept all the information transmitted among the corrupted entities in the network.
- *Dolev-Yao (DY) threat model:* The DY model [15] allows an adversary $\mathcal{F}$ not only to intercept the communicated messages, but also to modify, insert and erase the messages that are being transmitted between various agents in the network.
- *Canetti and Krawczyk's model:* This model known as the "CK-adversary model" [16] has all the capabilities of an adversary under the DY model. In addition, the CK-adversary model allows the adversary $\mathcal{F}$ to compromise the "secret keys, secret credentials, and session states through the session hijacking attacks". As a result, "leakage of the short term secrets can lead to disclose the session key and other secrets" [16].

The drones deployed in the flying zones can not be always monitored 24×7. Hence, there is a possibility to hijack or even physically capture some drones in the network. The adversary $\mathcal{F}$ can then take advantage of using the compromised drones to extract all the credentials stored in their memory using the "power analysis attacks" [17]. Moreover, apart from the traditional number theoretic attacks, $\mathcal{F}$ can also launch the quantum attacks.

III. Preliminaries

Let $\mathbb{N}$ be the "set of all natural numbers," $\mathbb{Z}$ be the "set of all integers," and for $v \in \mathbb{N}$ define $f = 2^v \in \mathbb{N}$. Assume p is a large prime such that $p \equiv 1 \pmod{2f}$ and the finite field $\mathbb{Z}/p\mathbb{Z}$ is represented as $\mathbb{Z}_p$ where $\mathbb{Z}_p = \{0, 1, \ldots, p-1\}$. Now, define $R = \mathbb{Z}[x]/\langle x^f + 1\rangle$, $R_p = \mathbb{Z}_p[x]/\langle x^f + 1\rangle$, $R_{p,k} \subset R_p$ such that $R_{p,k} = \{g(x) \in R_p$, where the degree of $g(x)$ is at most $f-1$, all the co-efficients of $g(x) \in [-k, k]\}$, with $0 < k \leq \frac{p-1}{2}$ and D_{32}^f, with $f \geq 512$, consists of all polynomials of degree at most $f-1$ that have all zero co-efficients except at most 32 co-efficients that are $+1$ or -1. In addition, we define $\tilde{H}$: $R_p \rightarrow R_{p,1}$, and H: $\{0,1\}^* \rightarrow D_{32}^f$ as two collision-resistance one way hash functions.

A. Computational Hard Problems

In this section, we define the following computational hard problems that are associated with the lattice-based cryptography.

1) Learning-With-Error (LWE): Assume that $s \in_R \mathbb{Z}_p^f$ be a fixed vector and χ be an error distribution over $\mathbb{Z}$. The "Learning-with-Error (LWE)" problem is define as follows [18]. Choose a vector $a \in \mathbb{Z}_p^f$ from a uniform distribution over $\mathbb{Z}_p^f$ and a number $e \in \mathbb{Z}$, and then compute $t = a.s + e \pmod{p}$. Given m $(\geq f)$ samples of the form: $(a_i, a_i.s + e_i \pmod{p})$, where $a_i \in \mathbb{Z}_p^f$ and $e_i \in \mathbb{Z}$ is the error, that is, sampled from the error distribution χ, recover unique random secret s, for all $i = 1, 2, \ldots, m$. Note that "if the secret s is sampled from the same error distribution as e, the hardness assumption of the LWE problem is still valid".

2) Ring-LWE: At present, the security of most of the lattice based multi-signature as well as aggregate signature schemes is based on the hardness of Ring-LWE problem. The Ring-LWE can be defined as follows [18]. Pick $a \in R$, $s \in_R R_p$ as a polynomial of degree at most $f-1$ and χ as an error distribution over R. Define $D_{s,\chi}$, the Ring-LWE distribution, whose outputs $(a, a.s + e \pmod{x^f + 1})$ are in $R_p \times R_p$, where $e \leftarrow_R \chi$. Choose $a_i \in R$, for all $i = 1, 2, \ldots, m$. This problem is associated to recover the secret $s \in_R R_p$ with the uniform distribution over $s \in_R R_p$ from m $(\geq f)$ samples of the form: $(a_i, a_i.s + e_i \pmod{x^f + 1})$, where $e_i \in R$ is the error with co-efficients sampled from the error distribution χ.

3) Super NTRU-Encrypt: It is well-known that the secret key size and the error size in the super NTRU encrypt problem are small with respect to standard LWE problem. To solve LWE problem with smaller secrets and smaller error is as hard as solving standard LWE problem. The super NTRU-encrypt problem is defined as follows [19]. Given $(a,\ t = (a.s + 2e \pmod{x^f + 1}))) \in R_p \times R_p$, where $a \in R_p$, $s \in \{0,1\}^n$ and $e \in_R \{0,1\}^n$, to recover s from (a, t).

B. Lower-Order and Higher-Order Bits

Let $Y \in [-(p-1)/2, (p-1)/2] \bigcap \mathbb{Z}$ and $k \in \mathbb{Z}^+$. Then, every Y can be uniquely expressed as follows [20]: $Y = (2k +$

$1)Y^1 + Y^0$, where $Y^0 \in [-k, k]$ and $Y^1 = (Y - Y^0)/(2k + 1)$. Here, Y^0 is the "lower-order bits of Y" and Y^1 are the "higher-order bits of Y". These higher and lower order bits are used in the proposed scheme.

C. Compression

We define the following compression technique $Compress$ as proposed in [20]. We are given p, f and k such that $2fk > p$, and two vectors, say $y \in R_p$ and $z \in R_{p,k}$, where the co-efficients of z are small. We can then replace z by a "much more compressed vector z' while keeping the higher-order bits of $y + z$ and $y + z'$ as the same". We have the following results: $z' \leftarrow Compress(y, z, p, k)$ where $z' \in R_{p,k}$ and $(y + z)^1 = (y + z')^1$.

D. Generalized Lattice-Based Signature Scheme

A generalized lattice-based signature scheme was proposed in [20]. It consists of a set of algorithms, namely: 1) KeyGen, 2) Sign, and 3) Verification. A trusted authority, called a key generation center (KGC), who generates the secret key as well as public key for the signer. Note that D_{32}^f, $f \geq 512$ consists of "all the polynomials of degree at most $f - 1$ that have all zero co-efficients except at most 32 co-efficients that are $+1$ or -1". We take $H : \{0, 1\}^* \rightarrow D_{32}^f$ as a "collision-resistance one way hash function".

1) $KeyGen(1^\lambda) \rightarrow (pk, sk)$: This algorithm runs by the KGC, and produces a pair of secret key and public key as follows.

- Pick $a \in_R R_p$, where a is chosen uniformly at random from R_p. Also, pick (s_1, s_2) which are uniformly taken at random from $R_{p,1}$, that is, $(s_1, s_2) \in_R R_{p,1} \times R_{p,1}$.
- Compute $t = a.s_1 + s_2 \pmod{x^f + 1}$.
- Set secret key as $sk = (s_1, s_2)$ and the corresponding public key as $pk = (a, t)$ for the signer.

After generating the public and private keys, the KGC sends the these key pair to the signer through a secure channel.

2) $Sign(sk, A, m) \rightarrow (sig_1, sig_2)$: The signer runs the Sign algorithm to create a signature on a message m using its own secret key sk as follows.

1) Choose $(y_1, y_2) \in_R R_{p,k} \times R_{p,k}$.
2) Compute $c = H(a.y_1 + y_2, m)$, $z_1 = s_1.c + y_1 \pmod{x}^f + 1$, and $z_2 = s_2.c + y_2 \pmod{x^f + 1}$.
3) If z_1 or $z_2 \notin R_{p,k-32}$, go to Step 1.
4) Output (z_1, z_2, c).

The created signature on m is (z_1, z_2, c). The signer then sends the signed message $(m, (z_1, z_2, c))$ to the verifier via public channel.

3) $Verification(sig_1, sig_2) \rightarrow (Valid/Invalid)$: After receiving the signed message $(m, (z_1, z_2, c))$, the verifier runs the Verification algorithm, which is given below.

- Check if z_1 or $z_2 \in R_{p,k-32}$. If it is not valid, reject the signature and return $Invalid$.
- Check if $c = H(a.z_1 + z_2 - t.c, m)$ holds or not. If it is not valid, reject the signature and return $Invalid$.
- Accept the signature as valid and return $Valid$.

TABLE I
NOTATIONS AND THEIR MEANINGS

Symbol	Description
CR	Trusted control room
GSS_j	Trusted j^{th} ground station server (GSS)
$DApp_j$	j^{th} drone application
Agg	An aggregator
DR_i	i^{th} drone or unmanned aerial vehicle (UAV)
CS_k	Semi-trusted k^{th} "cloud server in blockchain center"
P2P CS	A "Peer-to-Peer cloud servers network"
KGC	Trusted key generation center acting as control room
msk, mpk	Master secret key and public key of KGC, respectively
p	A large prime number
R_p	A finite field of the type: $Z_p[x]/ < x^f + 1 >$, where $Z_p = \{0, 1, \cdots, p-1\}$, f is the highest degree of the polynomial, and $p \equiv 1 \pmod{2f}$
k	A positive integer $< \sqrt{p}$
$R_{p,k}$	Sub-field of R_p such that all the co-efficients of each polynomial of this sub-field are in $\in [-k, k]$
ID_{Agg}	Identity of the the aggregator, Agg
i	A positive integer in $\{1, 2, \cdots, n\} \bigcup \{ID_{Agg}\}$
R_{p,k_i}	A sub-field of R_p such that all the co-efficients of each polynomial of this sub-field are in $\in [-k_i, k_i]$
D_{32}^f	A set of polynomials of degree $f-1$ that have all 0 co-efficients except at most 32 co-efficients that are either $+1$ or -1
k_i	A positive integer $< \sqrt{p}$
$(\overline{s_{i,1}^0}, \overline{s_{i,2}^0})$	Secret key of DR_i
t_i	Public key of DR_i
m_i	A message on which DR_i will sign
z_1^i, z_2^i	A pair of signatures of DR_i
$t_{ID_{Agg}}$	Public key of Agg
$(\overline{s_{ID_{Agg},1}^0}, \overline{s_{ID_{Agg},2}^0})$	Secret key of Agg
m'	A message on which Agg will sign
(z_1^{Agg}, z_2^{Agg})	A pair of signatures of Agg
$H(\cdot)$	A "collision-resistant cryptographic one-way hash function which maps an arbitrary string to an element in D_{32}^f of fixed length"
$\tilde{H}(\cdot)$	Another "collision-resistant cryptographic one-way hash function which maps from R_p to $R_{p,1}$"
CTS_X	"Current timestamp" generated by an entity X
ΔT	"Maximum allowable transmission delay"
$\|$	Data concatenation
$KeyGen$	Key generation algorithm
$LSSign$	Lattice based single message signature generation
$LSVer$	Lattice based single message signature verification
$LASign$	Lattice based aggregate signature generation
$LAVer$	Lattice based aggregate signature verification

IV. PROPOSED LATTICE-BASED AGGREGATE SIGNATURE SCHEME

This section describes the proposed lattice-based aggregate signature scheme. The notations and their descriptions are provided in Table I, which are useful for analysis and discussion of the proposed scheme.

The proposed scheme contains the five algorithms: a) **KeyGen**, b) **LSSign**, c) **LSVer**, d) **LASign** and e) **LAVer**. In the proposed scheme, there is a trusted authority (KGC), n number of signers and an aggregator. The KGC has the master secret key msk and the corresponding master public key mpk. Using msk and mpk, the KGC runs the $KeyGen$ algorithm in order to generate the secret key and the public key for each signer as well as for aggregator. After execution of the $KeyGen$ algorithm, the KGC sends the associated secret key and public key pair to each signer as well as aggregator. After receiving the secret key, each signer, being the ith signer, will sign a message m_i using its own secret key to generate the associated signature using the lattice-based $LSSign$ algorithm, and send the

generated signature to the aggregator. The aggregator then verifies all the received signatures of the n signers using the lattice-based $LSVer$ algorithm. If all the signatures are valid, then only the aggregator runs the lattice-based aggregate signature algorithm $LASign$ to generate the combined compact signature on all the verified signatures received from the signers by using its own secret key. Finally, the aggregator sends the associated aggregate signature to the verifier for signature verification.

The details of these algorithms are discussed below.

A. KeyGen

This algorithm is executed by the KGC using the following steps:

- The KGC chooses $(s_1, s_2) \in R_p \times R_p$ in such a way that the co-efficients of the polynomials s_1 and s_2 come from $[-(p-1)/2, (p-1)/2]$ and $a \in_R R_p$. The KGC then sets its master secret key msk and master public key mpk as (s_1, s_2) and a, respectively.
- The KGC selects $(n+1)$ distinct positive integers k_1, $k_2, \ldots, k_n$, and $k_{ID_{Agg}}$ from the range $[-(p-1)/2, (p-1)/2]$ in such a way that $k_i < \sqrt{p}$, for all $i = 1, 2, \ldots, n$, ID_{Agg} for the n signers and the aggregator, respectively. For the ith-signer, where $i \in \{1, 2, \ldots, n\} \bigcup \{ID_{Agg}\}$, then KGC computes $s_1 = (2k_i + 1)s^1_{i,1} + s^0_{i,1}$ and $s_2 = (2k_i + 1)s^1_{i,2} + s^0_{i,2}$, where the co-efficients of $s^0_{i,1}, s^0_{i,2}$ are from $[-k_i, k_i]$. Note that $s^1_{i,1}$ and $s^1_{i,2}$ are the higher order bits of s_1 and s_2, respectively.
- For each ith signer and the aggregator, where $i \in \{1, 2, \ldots, n\} \bigcup ID_{Agg}$, the KGC chooses a "one-way collision-resistance hash function" of the type: $\tilde{H} : R_p \longrightarrow R_{p,1}$, which maps the inputs $(s^0_{i,1}, s^0_{i,2})$ to the output $(\overline{s^0_{i,1}}, \overline{s^0_{i,2}})$ in $R_{p,1}$. The KGC then computes $t_i = a.\overline{s^0_{i,1}} + \overline{s^0_{i,2}} \pmod{x^f + 1}$ and generates the associated secret key $sk_i = (\overline{s^0_{i,1}}, \overline{s^0_{i,2}})$ and the public key $pk_i = t_i$ for each ith signer and the aggregator, respectively.

The Small Integer Solution (SIS) problem in the lattice-based setting consists of the following [21]:

- $SIS_{p,n,m,k}$*-distribution:* Pick a matrix A randomly from $Z_p^{n \times m}$ and a vector $s \leftarrow \{-k, -k+1, \ldots, 0, \ldots, k-1, k\}^m$, where $0 < k \leq \frac{p-1}{2}$. Generate the output (A, As).
- $SIS_{p,n,m,k}$*-search problem:* Given an output (A, t) of the $SIS_{p,n,m,k}$-distribution, find an element $s \leftarrow \{-k, -k+1, \ldots, 0, \ldots, k-1, k\}^m$ for which $As = t$ is satisfied.
- $SIS_{p,n,m,k}$*-decision problem:* Given a pair (A, t), there is a negligible probability to decide if it is from $SIS_{p,n,m,k}$-distribution or uniform distribution over $Z_p^{n \times m} \times Z_p^n$.

Based on the results reported in [21], we have the following claims:

Claim 1: If $k \geq p^{n/m}$, where $m = 2n$, for any $A \in Z_p^{n \times m}$ and a uniformly chosen $s \leftarrow \{-k, -k+1, \ldots, 0, \ldots, k-1, k\}^{2n}$, there exists another $s' \leftarrow \{-k, -k+1, \ldots, 0, \ldots, k-1, k\}^{2n}$ with $s' \neq s$, such that $As = As'$ is satisfied.

Claim 2: If $k < p^{n/m}$ with $m = 2n$, for any $A \in Z_p^{n \times m}$ and a uniformly chosen $s \leftarrow \{-k, -k+1, \ldots, 0, \ldots, k-1, k\}^{2n}$, there does not exist any different $s' \leftarrow \{-k, -k+1, \ldots, 0, \ldots, k-1, k\}^{2n}$ for which $As = As'$ is satisfied.

It is observed that the SIS problem classically reduces to the LWE problem [21]. Furthermore, the LWE problem also reduces to the Ring-LWE problem. Therefore, Claim 1 implies that in the Ring-LWE problem, if $k \geq p^{n/m}$ with $m = 2n$, that is, $k \geq \sqrt{p}$, the probability of randomly selecting a colliding element is very high. Claim 2 also implies that if $k < \sqrt{p}$, only one solution exits with high probability. Otherwise, when $\sqrt{p} < k \ll p$, solving worst-case lattice problem in the ideal lattice will be computationally hard problem. Based on the above results, it is worth to notice the following observations [20]: 1) if $k_i \geq \sqrt{p}$, the solution will not be unique, 2) if $\sqrt{p} < k_i \ll p$, solving worst-case lattice problem in the ideal lattice will be a computationally hard, and 3) if $k_i < \sqrt{p}$, only one solution exits with a high probability.

B. LSSign

Each ith signer has its own secret key and public key pair (sk_i, pk_i). Suppose the ith signer wants to generate the signature on a message $m_i \in \{0, 1\}^*$. The execution of the $LSSign$ signature generation algorithm by the signer has the following steps:

1) The ith signer first picks randomly y^i_1, y^i_2 from R_{p,k_i}, computes $c_i \leftarrow H(((a.y^i_1 + y^i_2) \,||m_i)^1)$, $z^i_1 \leftarrow \overline{s^0_{i,1}}c_i + 2y^i_1 \pmod{x^f + 1}$ and $z^i_2 \leftarrow \overline{s^0_{i,2}}c_i + 2y^i_2 \pmod{x^f + 1}$, and then sets $Z_i = (z^i_1, z^i_2)$ for all $i = 1, 2, \ldots, n$.
2) The ith signer checks whether z^i_1 or $z^i_2 \in R_{p,2k_i-32}$. If it is not so, go to Step 1. Otherwise, the signature together with the message m_i will be taken as $((Z_i, c_i), m_i)$, that is, $(((z^i_1, z^i_2), c_i), m_i)$.

C. LSVer

In order to check whether the signed message $(((z^i_1, z^i_2), c_i), m_i)$ of the ith signer is valid or not, the aggregator being a verifier, needs to execute the lattice-based single message signature verification algorithm, $LSVer$, with the following steps:

- The aggregator first checks whether z^i_1 or $z^i_2 \in R_{p,2k_i-32}$. If the verification fails, the signature is rejected.
- The aggregator computes

$$c^*_i \leftarrow H\left(\left(\left(\frac{az^i_1 + z^i_2 - c_i.t_i}{2}\right) ||m_i\right)^1\right) \quad (1)$$

and verifies if $c^*_i = c_i$ holds or not. If it holds, the aggregator accepts the signature on the message m_i as valid.

In LSVer, $c_i \leftarrow H(((a.y^i_1 + y^i_2) \,||m_i)^1)$, $z^i_1 \leftarrow \overline{s^0_{i,1}}c_i + 2y^i_1 \pmod{x^f + 1}$ and $z^i_2 \leftarrow \overline{s^0_{i,2}}c_i + 2y^i_2 \pmod{x^f + 1}$. It is noted that both z^i_1 and z^i_2 are the polynomials of degree at most $f - 1$ whose coefficients are from $[-(2k_i - 32), (2k_i - 32)]$. This is because from the rejection sampling algorithm [21], the signature turns out to be valid if sup-norms of z^i_1 and z^i_2 are less than or equal to $2k_i - 32$, that is, $||z^i_1||_\infty \leq (2k_i - 32)$ and $||z^i_2||_\infty \leq (2k_i - 32)$. Next, assume that an adversary changes the message m_i to m'_i and sends the signed message $(((z^i_1, z^i_2), c_i), m'_i)$ of the ith signer to the aggregator. In that case, the aggregator first checks whether z^i_1 and $z^i_2 \in R_{p,2k_i-32}$. If it is so, the aggregator proceeds to compute $c^{**}_1 = H(((\frac{az^i_1 + z^i_2 - c_i.t_i}{2})$

$||m_i')^1) = H(((a(\overline{s^0_{i,1}}c_i + 2y_1^i \pmod{x^f+1}) + (\overline{s^0_{i,2}}c_i + 2y_2^i \pmod{x^f+1}) - c_i.t_i)/2\ ||m_i')^1) = H(((a.y_1^i + y_2^i)\ ||m_i')^1)$. This implies that $c_i^{**} \neq c_i$. In other words, the signature will be rejected. The validity of the signature correctness proof in LSVer is further provided in Theorem 1.

D. LASign

After receiving the individual signatures $(((z_1^i, z_2^i), c_i), m_i)$ on the message m_i, $i = 1, 2, \ldots, n$, the aggregator verifies each associated signature. If all the associated signatures are valid, then only the aggregator executes the lattice-based aggregate signature generation algorithm, $LASign$, on the combined message $(m_1, m_2, \ldots, m_n)$ using the following steps:

- The aggregator first computes $m' = H((Z_1, c_1, m_1)\ ||\cdots||(Z_n, c_n, m_n))$, where $Z_i = (z_1^i, z_2^i)$.
- The aggregator then chooses randomly y_1^{Agg}, y_2^{Agg} from $R_{p,k_{ID_{Agg}}}$, and computes $c_{Agg} \leftarrow H(((a.y_1^{Agg} + y_2^{Agg})\ ||m')^1)$. After that the aggregator determines the associated aggregate signature by computing $z_1^{Agg} \leftarrow \overline{s^0_{ID_{Agg},1}}\, c_{Agg} + 2y_1^{Agg} \pmod{x^f+1}$ and $z_2^{Agg} \leftarrow \overline{s^0_{ID_{Agg},2}}\, c_{Agg} + 2y_2^{Agg} \pmod{x^f+1}$. Let $Z_{Agg} = (z_1^{Agg}, z_2^{Agg})$.
- The aggregator now verifies if z_1^{Agg} or $z_2^{Agg} \in R_{p,2k_{ID_{Agg}}-32}$. If it is not so, go to Step 1. Otherwise, the aggregate signature on the combined message $(m_1, m_2, \ldots, m_n)$ is treated as (Z_{Agg}, c_{Agg}). Finally, the aggregator sends the aggregate signature (Z_{Agg}, c_{Agg}) along with $\{(z_1^i, z_2^i, c_i, m_i)\ |i = 1, 2, \ldots, n\}$ to the verifier.

E. LAVer

After receiving the aggregate signature (Z_{Agg}, c_{Agg}) along with $\{(z_1^i, z_2^i, c_i, m_i)\ |i = 1, 2, \ldots, n\}$, the verifier executes the lattice-based aggregate signature verification algorithm, $LAVer$, with the following steps:

- The verifier sets $m^* = H((z_1^1, z_2^1, c_1, m_1)\ ||\cdots||(z_1^n, z_2^n, c_n, m_n))$.
- The verifier checks whether $z_1^{Agg}, z_2^{Agg} \in R_{p,2k_{ID_{Agg}}-32}$ or not. It the verification holds, the verifier calculates

$$c^*_{Agg} \leftarrow H\left(\left(\left(\frac{a.z_1^{Agg} + z_2^{Agg} - c_{Agg}.t_{ID_{Agg}}}{2}\right)||m^*\right)^1\right). \tag{2}$$

- The verifier validates if $c^*_{Agg} = c_{Agg}$. If it holds, the aggregate signature is valid.

It is noted that both z_1^{Agg} and z_2^{Agg} are the polynomials of degree at most $f-1$ whose coefficients are from $[-(2k_{ID_{Agg}} - 32), (2k_{ID_{Agg}} - 32)]$. From the rejection sampling algorithm [21], the signature becomes valid when the sup-norms of z_1^{Agg} and z_2^{Agg} are less than or equal to $2k_{ID_{Agg}} - 32$, that is, $||z_1^{Agg}||_\infty \leq (2k_{ID_{Agg}} - 32)$ and $||z_2^{Agg}||_\infty \leq (2k_{ID_{Agg}} - 32)$. In addition, the validity of the signature correctness proof in LAVer is further provided in Theorem 2.

V. Applying Lattice-Based Aggregate Signature Scheme in IoT-Enabled Drones Applications

This section describes how the designed lattice-based aggregate signature scheme mentioned in Section IV is applied for the real-time IoD applications.

With respect to the network model shown in Fig. 1, there are several entities in the network, like "drones," "ground station servers" (GSS), "control room" (CR) and "cloud servers" (CS). The n_{cs} cloud servers CS_l, $l = 1, 2, \ldots, n_{cs}$, form a "Peer-to-Peer (P2P) cloud servers (P2P CS) network," known as blockchain center (BC) which is mainly responsible to form the blocks of received transaction from the respective GSS for the drone applications and mine the blocks into the blockchain center BC through consensus algorithm. For a particular application, say $DApp_j$, the in-charge GSS_j will be responsible to collect the data from the drones which contain in that application. The control room (CR) is in-charge of registering the drones and GSS for each drone application $DApp_j$ in offline mode.

The involved phases are 1) "registration phase," 2) "data collection phase," 3) "data aggregation phase," 4) "blockchain implementation phase," and 5) "dynamic drone deployment phase". We assume that "the entities involved in the network are synchronized with their clocks," which is a reasonable assumption [22], [23], [24]. The time-stamping mechanism helps us to resist the replay attacks against an adversary. The notations tabulated in Table I are also used to describe these various phases.

A. Registration Phase

In this section, we discuss two types of registration processes of a drone (DR_i) and its associated GSS_j for a particular application $DApp_j$. For $DApp_j$, if n_{dr} number of drones need to be deployed, the control room (CR) will be responsible for registering them prior to their deployment. In a similar way, the in-charge GSS_j for $DApp_j$ needs to be also registered by the CR.

1) Drone Enrolment: To register a drone DR_i, $i = 1, 2, \ldots, n_{dr}$ in $DApp_j$, the CR acting as the KGC executes the $KeyGen$ algorithm as discussed in Section IV-A to generate the secret key $sk_i = (\overline{s^0_{i,1}}, \overline{s^0_{i,2}})$ and the corresponding public key $pk_i = t_i$. In addition, the CR also generates a unique identity ID_{DR_i} for DR_i. At the end, the CR pre-loads the information $\{ID_{DR_i}, (sk_i, pk_i)\}$ into the memory of DR_i before its deployment in $DApp_j$. The CR publishes the public key pk_i.

2) GSS Registration: In order to register GSS_j for the application $DApp_j$, GSS_j first selects an identity ID_{Agg_j} and sends it to the CR via a secure channel. Here, the CR acts as the KGC. Next, the CR runs the $KeyGen$ algorithm as discussed in Section IV-A to generate the secret key $sk_{Agg_j} = (\overline{s^0_{ID_{Agg_j},1}}, \overline{s^0_{ID_{Agg_j},2}})$ and the corresponding public key $pk_{Agg_j} = t_{ID_{Agg_j}}$, and sends $\{sk_{Agg_j}, pk_{Agg_j}\}$ securely to the GSS_j. Finally, the GSS_j stores $\{ID_{Agg_j}, (sk_{Agg_j}, pk_{Agg_j})\}$ in its secure database. The CR also publishes pk_{Agg_j} as public.

B. Data Collection Phase

In this phase, each drone DR_i in an application $DApp_j$ will collect the data (information) from its flying zone for a particular time period. The following steps are involved in this phase:

Step 1: Each DR_i generates a current timestamp CTS_{DR_i}, creates a message as $m_i = \{ID_{DR_i}, ID_{Agg_j}, (RTS_{start}, RTS_{end}), Data_{DR_i}, CTS_{DR_i}\}$ and runs the $LSSign$ signature generation algorithm stated in Section IV-B to create the signature, say $SSig_{m_i}$ on the message m_i using its own secret key $sk_i = (\overline{s^0_{i,1}}, \overline{s^0_{i,2}})$ as follows:

$$SSig_{m_i} = \left(\left(z_1^i, z_2^i\right), c_i\right). \tag{3}$$

Here, RTS_{start} and RTS_{end} denote the recorded start time and end time for the data $Data_{DR_i}$.

Step 2: DR_i then creates a transaction TX_{DR_i} of the form: $TX_{DR_i} = \{m_i, SSig_{m_i}, pk_i\}$ and sends the "data collection message" $Msg_{DR_i} = \{TX_{DR_i}\}$ to the corresponding GSS_j via a public channel.

C. Data Aggregation Phase

In this phase, the associated GSS_j residing in its application $DApp_j$ collects the transactions from its various deployed drones in the network. After that, the GSS_j being an aggregator Agg_j executes the following steps:

Step 1: Suppose GSS_j receives the "data collection message" $Msg_{DR_i} = \{TX_{DR_i}\}$ from the ith drone $(i = 1, 2, \ldots, n_{dr})$ in $DApp_j$, at time TS_{DR_i}. At first, the GSS_j checks the validity of the received timestamp CTS_{DR_i} by the condition: $|CTS_{DR_i} - TS_{DR_i}| < \Delta T$. If it is so, the message Msg_{DR_i} is treated as a fresh one. Otherwise, the phase is immediately discarded by the GSS_j.

Step 2: The GSS_j then runs the $LSVer$ signature verification algorithm stated in Section IV-C on the message m_i with the help of DR_i's public key pk_i $(i = 1, 2, \ldots, n_{dr})$. If the signature is valid, the GSS_j treats the received transaction TX_{DR_i} as valid.

Step 3: For all successfully validated t_n transactions, say $TX_{DR_{i_1}}, TX_{DR_{i_2}}, \ldots, TX_{DR_{i_{t_n}}}$, the GSS_j executes the $LASign$ aggregate signature generation algorithm stated in Section IV-D to create the aggregate signature $ASig_{Agg_j}$ on the aggregated data: $\{(Sig_{m_{i_1}}, m_{i_1}), (Sig_{m_{i_2}}, m_{i_2}), \ldots, (Sig_{m_{i_{t_n}}}, m_{i_{t_n}})\}$, with the help of its own secret key $sk_{Agg_j} = (\overline{s^0_{ID_{Agg_j},1}}, \overline{s^0_{ID_{Agg_j},2}})$, where $ASig_{Agg_j} = (Z_{Agg_j}, c_{Agg_j})$. Finally, the GSS_j sends the "data aggregation message" $Msg_{Agg_j} = \{(TX_{DR_{i_1}}, TX_{DR_{i_2}}, \ldots, TX_{DR_{i_{t_n}}}), ASig_{Agg_j}, pk_{Agg_j}\}$ to a cloud server, say CS_k, via open channel.

D. Blockchain Implementation Phase

This phase is executed by an in-charge cloud server CS_k, who receives the "data aggregation message" from its respective GSS. Suppose the cloud server CS_k in the P2P CS network receives $Msg_{Agg_j} = \{(TX_{DR_{i_1}}, TX_{DR_{i_2}}, \ldots, TX_{DR_{i_{t_n}}}), ASig_{Agg_j}\}$ from GSS_j. After that the following steps are executed:

Block Header	
Block Version	$BVer$
Previous Block Hash	PBH
Merkle Tree Root	MTR
Timestamp	TS
Owner of Block	$Agg\ (GSS)$
Owner Public Key	$pk_{ID_{Agg}}$
Block Payload	
List of t_n Transactions	$\{TX_{DR_l} \mid l = i_1, i_2, \cdots, i_{t_n}\}$
Aggregate Signature on t_n Transactions	$ASig_{Agg}$
Current Block Hash	$CBHash$

Fig. 2. Structure of a block.

Step 1: CS_k runs the aggregate signature verification algorithm, $LAVer$ stated in Section IV-E on $(m_l, SSig_{m_l})$, $(l = i_1, i_2, \ldots, i_{t_n})$, from the transactions $(TX_{DR_{i_1}}, TX_{DR_{i_2}}, \ldots, TX_{DR_{i_{t_n}}})$. If the aggregate signature validation is successful, CS_k executes the next step; otherwise, this phase is discarded.

Step 2: CS_k now forms a block, say, $Block$ as shown in Fig. 2 containing the transactions $(TX_{DR_{i_1}}, TX_{DR_{i_2}}, \ldots, TX_{DR_{i_{t_n}}})$, their aggregate signature $ASig_{Agg_j}$, public key of the aggregator $pk_{Agg_j}\}$, unique block version $(BVer)$, Merkle tree root (MTR), hash of the previous block in the chain (PBH), block creation timestamp (TS), owner of the block and current block hash $(CBHash)$. Note that for the blockchain purpose, we use the "Secure Hash Algorithm (SHA-256)" that maps any arbitrary string to 256-bit hash output. The Merkle tree root is created on all the t_n transactions $(TX_{DR_{i_1}}, TX_{DR_{i_2}}, \ldots, TX_{DR_{i_{t_n}}})$ present in the block. The current block hash is computed by hashing all the fields containing in the block as $CBHash = Hash(\text{Block Header}||\text{Block Payload})$, where $Hash(\cdot)$ is SHA-256 hash function.

Step 3: CS_k then runs a leader selection algorithm as stated in [25] to pick a leader from the P2P CS network. The following voting procedure is used to elect the leader from the cloud servers CS_k $(k = 1, 2, \ldots, n_{cs})$ present in the P2P CS network:

- Initially, all CS_k's are considered as the followers and voting participants.
- Set a time threshold for finishing the voting process and also start the voting process.
- A follower declares itself as a candidate in the voting process and sends a request to other followers for a vote. It then waits for the reply votes from the peer nodes.
- After receiving the reply votes from other followers, the candidate checks whether the number of valid votes reaches to a pre-defined threshold value for winning the vote or not.
- The candidate is declared as the winner if its threshold value reaches to the required number of valid votes for winning the vote prior to the voting process is finished.

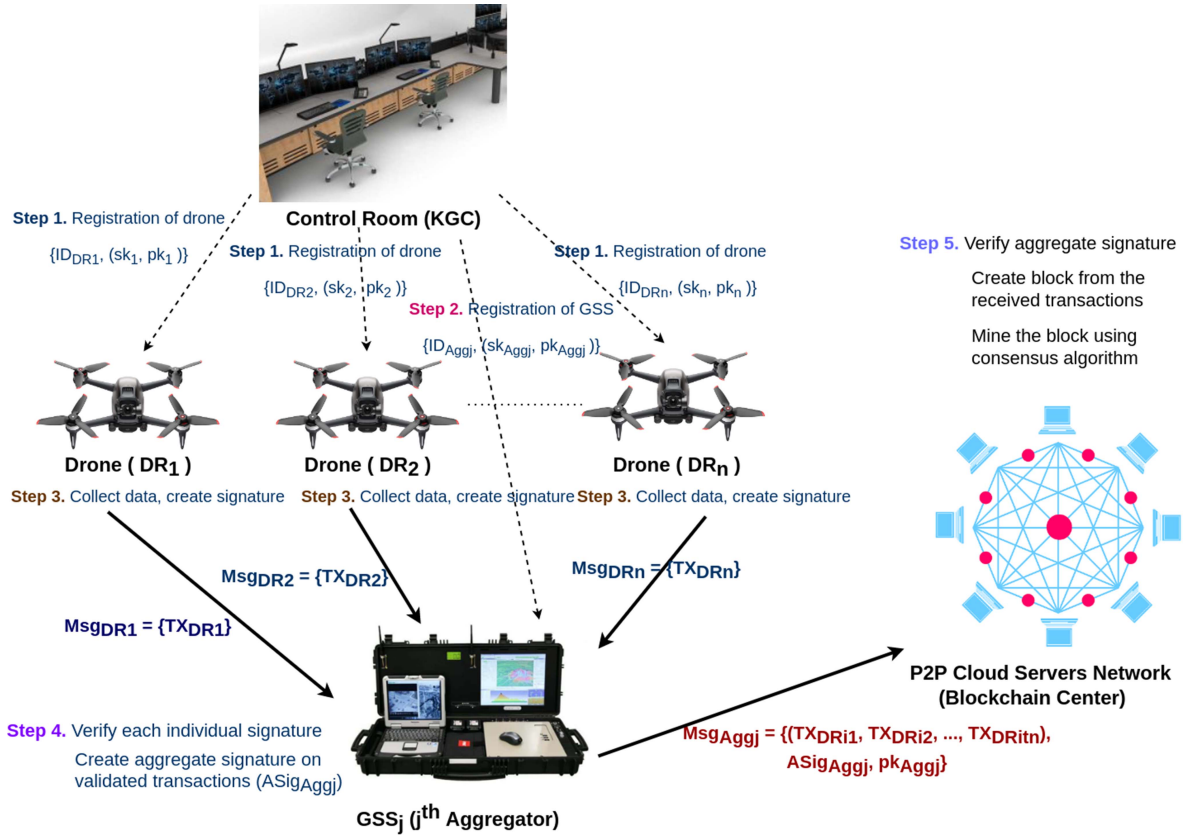


Fig. 3. Lattice-based aggregate signature scheme in action for an IoT-enabled drones application, $DApp_j$, using blockchain.

- Once the leader is elected, it broadcast a message to the entire P2P network for the block mining process.
- Similarly, other follower nodes can be elected as the leaders for the next rounds of voting.

Step 4: Assume that L is elected as the leader among the cloud servers CS_k, $(k = 1, 2, \ldots, n_{cs})$. L then executes a voting-based consensus algorithm for verifying and mining the created block into the blockchain. For this purpose, we apply the existing "Practical Byzantine fault tolerance (PBFT)" [26] consensus algorithm. The steps behind the consensus process are given below:

- L broadcasts the block, $Block$, among all its peer nodes in the network with a voting request.
- Each peer node then verifies the received block $Block$ by means of verifying the Merkle tree root, aggregate signature and current block hash on all the transactions present in that block. If all the verifications pass successfully, $Block$ is treated to be valid one and the concerned peer node will send a voting reply message with verification status as "valid".
- L maintains a counter, Ctr_L which is initialized to 0. For all the valid voting reply messages from the peer nodes, L increment Ctr_L by 1, that is, $Ctr_L = Ctr_L + 1$. Now, L checks if Ctr_L exceeds more than a pre-defined threshold value, $2 * f_{cs} + 1$, where f_{cs} denotes the number of faulty nodes out of n_{cs} nodes in the P2P CS network. If it is so, L will send a "commit response" message to its all (peer) follower nodes and add the block $Block$ in its ledger. After receiving the "commit response" message from L, other peer nodes also add the same block $Block$ in their respective ledgers.

The overall scenario of the messages flow among various entities is illustrated in Fig. 3.

E. Dynamic Drone Deployment Phase

An existing drone may be physically compromised according to the threat model stated in Section II-B, because it may not be possible to monitor 24 × 7 all the drones flying in the zones or some drones may malfunction. Thus, it becomes to add some new drones in the network after their initial deployment.

If a new drone DR_i^{new} needs to be deployed in an existing drone application, say $DApp_j$, the CR acting as the KGC will execute the $KeyGen$ algorithm as discussed in Section IV-A in order to create the secret key $sk_i^{new} = (\overline{s_{i,1}^{0,new}}, \overline{s_{i,2}^{0,new}})$ and the corresponding public key $pk_i^{new} = t_i^{new}$. Next, the CR generates a unique identity $ID_{DR_i}^{new}$ for the deployed DR_i^{new}. Finally, the CR stores the information $\{ID_{DR_i}^{new}, (sk_i^{new}, pk_i^{new})\}$ into the memory of DR_i^{new} and publishes the public key pk_i^{new}.

VI. Security Analysis

In this section, we first provide the correctness proof of the signature verification. Next, through the formal security analysis under the standard model we prove the unforgeability of the proposed lattice-based aggregate signature scheme presented in

Section IV. Finally, through the informal (heuristic) security analysis we also show that the proposed lattice-based aggregate signature scheme applied in IoT-enabled drones applications is robust against "potential attacks against a passive or an active adversary".

A. Correctness Proof

In the following, we proof the correctness of signature verifications of both the single signature and aggregate signature stated in Sections IV-C and IV-E, respectively.

Theorem 1: In the proposed lattice-based single signature verification stated in Section IV-C, if the condition $c_i^* = c_i$ is satisfied, the signature is valid.

Proof: We have the signature (Z_i, c_i) on a message m_i, where $Z_i = (z_1^i, z_2^i)$, $c_i \leftarrow H(((a.y_1^i + y_2^i)\,||m_i)^1)$, $z_1^i \leftarrow \overline{s_{i,1}^0}c_i + 2y_1^i \pmod{x^f+1}$ and $z_2^i \leftarrow \overline{s_{i,2}^0}c_i + 2y_2^i \pmod{x^f+1}$. After receiving the signed message, the aggregator computes $c_i^* \leftarrow H(((\frac{az_1^i + z_2^i - c_i.t_i}{2})\,||m_i)^1)$.

To show $c_i^* = c_i$, it suffices to show that $H(((a.y_1^i + y_2^i)\,||m_i)^1) = H(((\frac{az_1^i+z_2^i-c_i.t_i}{2})\,||m_i)^1)$, that is, $a.y_1^i + y_2^i = \frac{az_1^i+z_2^i-c_i.t_i}{2}$. Now, $az_1^i + z_2^i - c_i.t_i = a(\overline{s_{i,1}^0}c_i + 2y_1^i) + (\overline{s_{i,2}^0}c_i + 2y_2^i) - c_i(a.\overline{s_{i,1}^0} + \overline{s_{i,2}^0}) = 2ay_1^i + 2y_2^i = 2(ay_1^i + y_2^i)$. Thus, $\frac{az_1^i+z_2^i-c_i.t_i}{2} = ay_1^i + y_2^i$. Hence, the signature is valid. □

Theorem 2: In the proposed lattice-based aggregate signature verification stated in Section IV-D, if the condition $c_{Agg}^* = c_{Agg}$ is satisfied, the signature is valid.

Proof: We have $c_{Agg} \leftarrow H(((a.y_1^{Agg} + y_2^{Agg})\;||m')^1)$ and $c_{Agg}^* \leftarrow H(((\frac{a.z_1^{Agg}+z_1^{Agg}-c_{Agg}.t_{ID_{Agg}}}{2})\;||m^*)^1)$. To show $c_{Agg}^* = c_{Agg}$, it is sufficient to prove that $\frac{a.z_1^{Agg}+z_2^{Agg}-c_{Agg}.t_{ID_{Agg}}}{2} = a.y_1^{Agg} + y_2^{Agg}$, and $m' = m^*$. Now,

$$\begin{aligned} a.z_1^{Agg} + z_2^{Agg} - c_{Agg}.t_{ID_{Agg}} &= a\overline{s_{ID_{Agg},1}^0}c_{Agg} + 2ay_1^{Agg} \\ &+ \overline{s_{ID_{Agg},2}^0}c_{Agg} + 2y_2^{Agg} \\ &- c_{Agg}\left(a.\overline{s_{ID_{Agg},1}^0} + \overline{s_{ID_{Agg},2}^0}\right) \\ &= 2\left(ay_1^{Agg} + y_2^{Agg}\right). \end{aligned} \tag{4}$$

This means that $\frac{a.z_1^{Agg}+z_2^{Agg}-c_{Agg}.t_{ID_{Agg}}}{2} = a.y_1^{Agg} + y_2^{Agg}$. Again, $m^* = H((z_1^1, z_2^1, c_1, m_1)\;||\cdots||(z_1^n, z_2^n, c_n, m_n)) = H((Z_1, c_1, m_1)\;||\cdots||(Z_n, c_n, m_n)) = m'$. Hence, the aggregate signature is valid. □

B. Formal Security Analysis

We say that the proposed lattice-based aggregated signature scheme (LBAS) is unforgeable in the random oracle model under the condition that the Ring-LWE must be a computationally hard problem. In Theorem 1, we prove that LBAS is unforgeable against an adversary based on the hardness of the computational Ring-LWE problem defined in Section III-A2. Here, we apply the generalized forking lemma [27].

Theorem 3: Assume that there exists an adversary (forger) $\mathcal{F}$ who has a non-negligible probability, say ϵ_F, in breaking our lattice-based aggregated signature scheme, LBAS $= (KeyGen, LSSign, LSVer, LASign, LAVer)$. Then, there exists a simulator $\mathcal{C}$ who can solve an instance of the Ring-LWE problem, with a non-negligible advantage $\frac{\epsilon_F}{8q_H}$, where q_H denotes the total number of permissible hash queries.

Proof: Assume that the adversary $\mathcal{F}$ can win the unforgeable game played with the simulator $\mathcal{C}$. Let the given Ring-LWE instance be $A = (a, 1) \in R_p \times \{1\}, p \equiv 1 \pmod 2f), f = \mathcal{O}(\lambda)$, λ is the security parameter. $\mathcal{C}$ then sets the public parameters pp in the form: $pp = (f,\ p,\ a,\ k_1,\ k_2, \ldots, k_n,\ k_{ID_{Agg}},\ \tilde{H},\ H)$, where $k_i < \sqrt{p}$, for all $i = 1, 2, \ldots, n, ID_{Agg}$, and $\tilde{H}$: $R_p \longrightarrow R_{p,1}$ and H: $\{0,1\}^* \rightarrow D_{32}^f$ are two cryptographically secure collision resistance hash functions.

The simulator $\mathcal{C}$ chooses two random polynomials $(s_1^{dr}, s_2^{dr}) \in R_p \times R_p$ for a drone DR_i such that $s_1^{dr} = (2k_i + 1)(s_1^*)^i + (s_1^{**})^i$ and $s_2^{dr} = (2k_i + 1)(s_2^*)^i + (s_2^{**})^i$. Similarly, $\mathcal{C}$ chooses two random polynomials $(s_1^{agg}, s_2^{agg}) \in R_p \times R_p$ for GSS_j so that $s_1^{agg} = (2k_{ID_{Agg}} + 1)s_{ID_{Agg},1}^* + s_{ID_{Agg},1}^{**}$ and $s_2^{agg} = (2k_{ID_{Agg}} + 1)s_{ID_{Agg},2}^* + s_{ID_{Agg},2}^{**}$. Now, $\tilde{H}((s_1^{**})^i) = (\overline{s_1^{**}})^i$, $\tilde{H}((s_2^{**})^i) = (\overline{s_2^{**}})^i$, and $\tilde{H}(s_{ID_{Agg},1}^{**}) = \overline{s_{ID_{Agg},1}^{**}}$, $\tilde{H}(s_{ID_{Agg},2}^{**}) = \overline{s_{ID_{Agg},2}^{**}}$, where $((\overline{s_1^{**}})^i, (\overline{s_2^{**}})^i) \in R_{p,1} \times R_{p,1}$ and $(\overline{s_{ID_{Agg},1}^{**}}, \overline{s_{ID_{Agg},2}^{**}}) \in R_{p,1} \times R_{p,1}$. After that $\mathcal{C}$ computes $t_i^* = a.(\overline{s_1^{**}})^i + (\overline{s_2^{**}})^i$ and $t_{ID_{Agg}}^* = a.\overline{s_{ID_{Agg},1}^{**}} + \overline{s_{ID_{Agg},2}^{**}}$.

$\mathcal{C}$ then selects a random coin, say $\nabla = (\partial, W)$, where $W = \{y_1^*, y_2^*, \ldots, y_{q_H}^*\}$, where each $y_i^* \in R_{p,k_i} \times R_{p,k_i}$ for all $i = 1, 2, \ldots, q_H, \partial \in R_p$, and y_i^* is of the form: $y_i^* = ((y_1^i)^*, (y_2^i)^*)$. $\mathcal{C}$ guesses an index l such that the $\mathcal{F}$ can forge y_l^* involved in the lth hash query. The $\mathcal{C}$ executes on the inputs t_i^*, $t_{ID_{Agg}}^*$, pp and ∇ to simulate the queries of $\mathcal{F}$ in the following way. Note that $\mathcal{F}$ is to permitted to query as many as n hash queries as well as n sign queries.

H-query: $\mathcal{C}$ first receives a hash H query on a message, say m_i. $\mathcal{C}$ maintains a list, say L_H containing of the elements of the form: $\{(m_i, H((A.y_i^*||m_i)^1)) : 1 \le j \le q_H\}$. If the message m_i, $1 \le i \le q_H$, is queried before, $\mathcal{C}$ will return it from the list L_H. Otherwise, if it is queried for the first time, $\mathcal{C}$ needs to honestly generate and return $H((A.y_i^*||m_i)^1)$ to the $\mathcal{F}$. After that $\mathcal{C}$ stores $(m_i, H((A.y_i^*||m_i)^1))$ in L_H.

Signature Generation Query: When the $\mathcal{F}$ queries the ith signature query on the message m_i, where $1 \le i \le q_S$ with $q_H = q_S$, the simulator $\mathcal{C}$ first needs to validate if the pair $(m_i, c_i^*) \in L_H$ exists or not, where $c_i^* = H((A.y_i^*||m_i)^1)$. If it does not exist or $i = l$, for $l \in \{1, 2, \ldots, q_S\}$, is guessed at the starting point of the game, then it is aborted; else, $\mathcal{C}$ will honestly create the signature Z_i^* for the honest signer i on the message m_i as follows:

i) Pick $y_i^* \in R_{p,k_i} \times R_{p,k_i}$.
ii) Calculate $c_i^* = H((A.y_i^*||m_i)^1)$.
iii) Compute $Z_i^* = ((\overline{s_1^{**}})^i,\ (\overline{s_2^{**}})^i)c_i^* + 2y_i^*$. If $Z_i^* \notin R_{p,2k_i-32} \times R_{p,2k_i-32}$, repeat from Step (i). Otherwise, return $\sigma_{m_i} = (Z_i^*, c_i^*, m_i)$ to the $\mathcal{F}$.

Aggregated H-query: $\mathcal{F}$ queries a hash-query on the message m^{**}. The simulator $\mathcal{C}$ first calculates $m' = H((Z_1^*, c_1^*, m_1)\;||\cdots||(Z_n^*, c_n^*, m_n))$, where $Z_i^* = ((\overline{s_1^{**}})^i, (\overline{s_2^{**}})^i)c_i^* + 2y_i^*$, and then checks if $m' = m^{**}$ holds or not. If it is not so, $\mathcal{C}$ aborts the

message m^{**}. On the other side, the $\mathcal{C}$ verifies it in the list L_H. If it is queried before, $\mathcal{C}$ returns it from the list L_H. If $ID_{Agg} = l$, where $l \in \{1, 2, \ldots, q_S\}$ is guessed at the staring of the game, then it is aborted; otherwise, $\mathcal{C}$ picks $y^*_{Agg} \in R_{p,k_{ID_{Agg}}} \times R_{p,k_{ID_{Agg}}}$ and calculates c^*_{Agg} as $c^*_{Agg} = H((A.y^*_{Agg} ||m^{**})^1)$.

Aggregated Signature Generation Query: When $\mathcal{F}$ queries an aggregated signature query on the message m^{**} to $\mathcal{C}$, the simulator $\mathcal{C}$ verifies if m^{**} is an authenticated valid message or not. Such a verification is performed using the following steps:

i) At first, $\mathcal{C}$ computes $H((Z^*_1, c^*_1, m_1) \,||\cdots||(Z^*_n, c^*_n, m_n))$ and checks if it matches with m^{**}.
ii) If the above verification does not hold, the $\mathcal{C}$ rejects m^{**}.
iii) Otherwise, $\mathcal{C}$ generates the aggregated signature as follows:

a) Choose $y^*_{Agg} \in R_{p,k_{ID_{Agg}}} \times R_{p,k_{ID_{Agg}}}$.
b) Generate $c^*_{Agg} = H((A.y^*_{Agg}||m^{**})^1)$ and compute $Z^*_{Agg} = (\overline{s^{**}_{ID_{Agg},1}}, \overline{s^{**}_{ID_{Agg},2}}).c^*_{Agg} + 2y^*_{Agg}$.
c) The simulator $\mathcal{C}$ verifies if $Z^*_{Agg} \in R_{p,2k_{ID_{Agg}}-32} \times R_{p,2k_{ID_{Agg}}-32}$ or not. If it is not so, start again from Step (a). Otherwise, $\mathcal{C}$ sends $\sigma_{m^{**}} = (Z^*_{Agg}, c^*_{Agg}, m^{**}, t^*_{ID_{Agg}})$ to $\mathcal{F}$.

Now, $\mathcal{F}$ outputs $\sigma_{m^{**}}$ on the message m^{**}. If it is a valid forgery aggregate signature, then

$$\mathcal{F}c^{**}_{Agg} \leftarrow H\left(\left(\left(\frac{A.Z^*_{Agg} - c^*_{Agg}t^*_{ID_{Agg}}}{2}\right)||m^{**}\right)^1\right) \quad (5)$$

and checks if $c^{**}_{Agg} = c^*_{Agg}$ holds or not. On the other hand, the simulator returns "fail" if 1) $\sigma_{m^{**}}$ is not a valid forgery and 2) $c^*_{Agg} = c^*_l$. If the individual signature σ_{m_i} on the message m_i is valid, $Z^*_i \in R_{p,2k_i-32} \times R_{p,2k_i-32}$ holds. Thus, $c^*_i \leftarrow H(((\frac{A.Z^*_i - c^*_i t^*_i}{2})||m_i)^1)$ for all $i \in \{1, 2, \ldots, q_H\}$. The simulator $\mathcal{C}$ calculates $c^*_{Agg} \leftarrow H(((\frac{AZ^*_{Agg} - c^*_{Agg}t^*_{ID_{Agg}}}{2})||m^{**})^1)$, where $H((Z^*_1, c^*_1, m_1) \,||\cdots||(Z^*_n, c^*_n, m_n)) = m^{**}$ and c^*_is are simulated as in the H-query on the set $\{y^*_1, y^*_2, \ldots, y^*_{q_H}\}$. Let $U_i = H((A.y^*_i||m_i))$ for $i \in \{1, 2, \ldots, q_H\}$. The simulator $\mathcal{C}$ takes the inputs as the Rign-LWE instance A, random ∇ and $\mathcal{F}$ as a subroutine, and then outputs $\{Z^*_{Agg}, c^*_{Agg}, m^{**}, t^*_{ID_{Agg}}, U_1, U_2, \ldots, U_{q_H}\}$.

An algorithm, say D' is now constructed, which has the input the above defined simulator $\mathcal{C}$ and the Ring-LWE instance A, which solves the Ring-LWE problem. The algorithm D' uses the generalized forking lemma as stated in [27] and produces an output which becomes a solution of the Ring-LWE instance. Algorithm D' returns "fail" if the simulator $\mathcal{C}$'s output is $(0, \perp)$. Let the output of the $\mathcal{C}$ be (Out_1, Out_2), where $Out_1 = \{Z^*_{Agg}, c^*_{Agg}, m^{**}, t^*_{ID_{Agg}}, U_1, U_2, \ldots, U_{q_H}\}$ and $Out_2 = \{Z'_{Agg}, c'_{Agg}, m^{**}, t'_{ID_{Agg}}, U'_1, U'_2, \ldots, U'_{q_H}\}$. The following two outcomes are generated by the simulator $\mathcal{C}$ using two random ∇_1 and and ∇_2, respectively, such that $\nabla_1 = (\partial, y^*_1, y^*_2, \cdots y^*_t, y^*_{t+1}, \ldots, y^*_{q_H})$ and $\nabla_2 = (\partial, y^*_1, y^*_2, \ldots, y^*_t, y'_{t+1}, \ldots, y'_{q_H})$. The algorithm D' extracts Z'_{Agg} and Z^*_{Agg} from ∇_1 and ∇_2, respectively. Now, $Z'_{Agg} = Z^*_{Agg}$ implies that $(\overline{s^{**}_{ID_{Agg},1}}, \overline{s^{**}_{ID_{Agg},2}}).c^*_{Agg} + 2(y^*_{Agg,1}, y^*_{Agg,2}) = (\overline{s^{**}_{ID_{Agg},1}}, \overline{s^{**}_{ID_{Agg},2}}).c'_{Agg} + 2(y'_{Agg,1}, y'_{Agg,2})$. This means that $\overline{s^{**}_{ID_{Agg},1}}.c^*_{Agg} + 2y^*_{Agg,1} = \overline{s^{**}_{ID_{Agg},1}}.c'_{Agg} + 2y'_{Agg,1}$. Thus, $\overline{s^{**}_{ID_{Agg},1}}.(c^*_{Agg} - c'_{Agg}) + 2(y^*_{Agg,1} - y'_{Agg,1}) = 0$, or, $\overline{s^{**}_{ID_{Agg},2}}.c^*_{Agg} + 2y^*_{Agg,2} = \overline{s^{**}_{ID_{Agg},2}}.c'_{Agg} + 2y'_{Agg,2}$, or, $\overline{s^{**}_{ID_{Agg},2}}.(c^*_{Agg} - c'_{Agg}) + 2(y^*_{Agg,2} - y'_{Agg,2}) = 0$. Hence, $\overline{s^{**}_{ID_{Agg},1}} = -2(y^*_{Agg,1} - y'_{Agg,1}).(c^*_{Agg} - c'_{Agg})^{-1}$ and $\overline{s^{**}_{ID_{Agg},2}} = -2(y^*_{Agg,2} - y'_{Agg,2}).(c^*_{Agg} - c'_{Agg})^{-1}$. As a result, if the adversary $\mathcal{F}$ can compute $(y^*_{Agg,1} - y'_{Agg,1})$ and $(y^*_{Agg,2} - y'_{Agg,2})$, the adversary $\mathcal{F}$ will be to solve the Ring-LWE problem.

The simulator $\mathcal{C}$ can succeed in the game if it can guess the index l correctly as well as the forger $\mathcal{F}$ produces a valid forgery aggregate signature. This means that the success probability of the $\mathcal{C}$ turns out to be $\frac{\epsilon_F}{q_H}$. Assume that the time taken by the $\mathcal{F}$ to generate a valid forgery aggregate signature is denoted by $t_{\mathcal{F}}$. It is worth to notice that the total running time of $\mathcal{F}$ depends on its running time plus the time needed by the $\mathcal{C}$ for responding the queries like hash and sign queries. Then, $\mathcal{C}$'s running time becomes $t'_{\mathcal{F}} = t_{\mathcal{F}} + t_{qH} + t_{qS}$, where t_{qH} and t_{qS} are respectively the time needed for hash and sign queries. With the help of the generalized forking lemma as stated in [27], if we take $p > \frac{8q_H}{\epsilon_F}$, the running time of the simulator $\mathcal{C}$ becomes $\frac{t'_{\mathcal{F}}.8q^2_H}{\epsilon_F.\log_e(\frac{8q_H}{\epsilon_F})}$ with the success probability (advantage) of at least $\frac{\epsilon_F}{8q_H}$. Hence, the theorem follows. □

C. Informal Security Analysis

In the following we show that the proposed scheme is robust against the following potential attacks needed in an IoT-enabled drones environment.

1) Replay Attack: Suppose an adversary $\mathcal{F}$ intercepts the "data collection message" $Msg_{DR_i} = \{TX_{DR_i}\}$ during the data collection phase stated in Section V-B that is sent from a drone DR_i to the corresponding GSS_j via a public channel, where $TX_{DR_i} = \{m_i, SSig_{m_i}, pk_i\}$, $m_i = \{ID_{DR_i}, ID_{Agg_j}, (RTS_{start}, RTS_{end}), Data_{DR_i}, CTS_{DR_i}\}$ and $SSig_{m_i} = ((z^i_1, z^i_2), c_i)$. Now, if $\mathcal{F}$ resends the same message to the recipient GSS_j after some time, the GSS_j will check the validity of the received timestamp CTS_{DR_i}. If it does not hold, the GSS_j will discard the message and it will be treated as old message. Thus, the timestamping mechanism helps in achieving the protection against replay attack.

2) Man-in-the-Middle Attack: In this attack, the adversary $\mathcal{F}$ will intercept the "data collection message" $Msg_{DR_i} = \{TX_{DR_i}\}$ during the data collection phase and try to modify the message so that the recipient GSS_j will not be aware of the modified valid message. In order to do so, $\mathcal{F}$ can create a current timestamp $fCTS_{DR_i}$ and fake data $fData_{DR_i}$ to compute $m^f_i = \{ID_{DR_i}, ID_{Agg_j}, (RTS_{start}, RTS_{end}), fData_{DR_i}, fCTS_{DR_i}\}$. However, $\mathcal{F}$ can not compute the signature $SSig^f_{m_i}$ on the modified message m^f_i on behalf of the drone DR_i because the secret key sk_i of DR_i is unknown to $\mathcal{F}$. Thus, $\mathcal{F}$ can not send the modified message $fMsg_{DR_i} = \{fTX_{DR_i}\}$, where $TX_{DR_i} = \{m^f_i, SSig^f_{m_i}, pk_i\}$. A similar scenario happens when $\mathcal{F}$ will try to modify the "data aggregation message" $Msg_{Agg_j} = \{(TX_{DR_{i_1}}, TX_{DR_{i_2}}, \ldots, TX_{DR_{i_{tn}}}), ASig_{Agg_j},$

$pk_{Agg_j}\}$ to send to a cloud server, CS_k, on behalf of GGS_j due to generation of the aggregate signature $ASig_{Agg_j}$. This restricts $\mathcal{F}$ to launch the man-in-the-middle attack.

3) Impersonation Attacks: In this case, we consider two types of impersonation attacks: a) "drone impersonation attack" and b) "GSS impersonation attack".

- In "drone impersonation attack," we assume that the adversary $\mathcal{F}$ wants to impersonate the GSS_j on behalf of a legal drone DR_i. To do so, $\mathcal{F}$ can generate a current timestamp $ICTS_{DR_i}$ and fake impersonated data $IData_{DR_i}$ to compute $m_i^I = \{ID_{DR_i}, ID_{Agg_j}, (RTS_{start}, RTS_{end}), IData_{DR_i}, ICTS_{DR_i}\}$. However, $\mathcal{F}$ will stuck in computing the signature $SSig_{m_i}^I$ on the message m_i^I on behalf of the drone DR_i because the secret key sk_i of DR_i is not available to $\mathcal{F}$. Thus, $\mathcal{F}$ can not create a valid message of the type: $Msg_{DR_i} = \{TX_{DR_i}\}$ and send it to the GSS_j. As a result, "drone impersonation attack" is not possible in the proposed scheme.
- In "GSS impersonation attack," suppose the adversary $\mathcal{F}$ wants to impersonate a cloud server CS_k on behalf of its associated GSS_j. For fulfilling this purpose, $\mathcal{F}$ needs to create an impersonated "data aggregation message" as $Msg_{Agg_j} = \{(TX_{DR_{i_1}}, TX_{DR_{i_2}}, \ldots, TX_{DR_{i_{tn}}}), ASig_{Agg_j}, pk_{Agg_j}\}$. For this purpose, $\mathcal{F}$ will have the transactions $(TX_{DR_{i_1}}, TX_{DR_{i_2}}, \ldots, TX_{DR_{i_{tn}}})$ which contain the individual signature created by the respective drones DR_i. However, to create a new aggregate signature, $\mathcal{F}$ needs to have the secret key $sk_j = (\overline{s^0_{ID_{Agg_j},1}}, \overline{s^0_{ID_{Agg_j},2}})$. This restricts $\mathcal{F}$ to generate the impersonated "data aggregation message" as Msg_{Agg_j} on behalf of the GSS_j to send it the cloud server CS_k. As a result, "GSS impersonation attack" is also resisted in the proposed scheme.

4) Physical Drone Capture Attack: According to the threat model stated in Section II-B, a registered drone, say DR_i, can be physically captured by the adversary $\mathcal{F}$. Using the "power analysis attacks" [17], $\mathcal{F}$ can easily extract all the credentials stored in its memory. Thus, the adversary $\mathcal{F}$ has the credentials $\{ID_{DR_i}, (sk_i, pk_i)\}$. Note that during the $KeyGen$ phase stated in Section IV-A all the drones are provided with the distinct and unique identities and secret-public keys pairs. As a result, $\mathcal{F}$ can only create valid "data collection message" on behalf of the compromised DR_i. However, $\mathcal{F}$ can not generate any valid "data collection message" on behalf of the remaining non-compromised drones in the network as their secret keys are unknown to the adversary $\mathcal{F}$. This shows that the proposed scheme is "unconditionally secure against drone capture attack".

5) Quantum Attacks: The hybrid lattice-reduction is considered as an attack where an adversary requires to solve the "shortest vector problem (SVP)" which is explained as follows. We are given a basis of vectors of a lattice where the vectors are the fixed-length tuples of integers. We need to determine a non-zero vector whose length is the shortest vector's length.

Now, when the "hybrid lattice-reduction" and "meet-in-the middle (MiTM)" attacks are combined together, a hybrid attack is then formed. This attack is an important attack for evaluation of the security of many lattice-based cryptographic schemes, such as NTRU (NTRUEncrypt and NTRUSign). In case of a generic attack, it is refereed to an attack where a secret key is needed to be recovered by the adversary based on generation of the decryption errors which is a treated as a "chosen-ciphertext attack (CCA)".

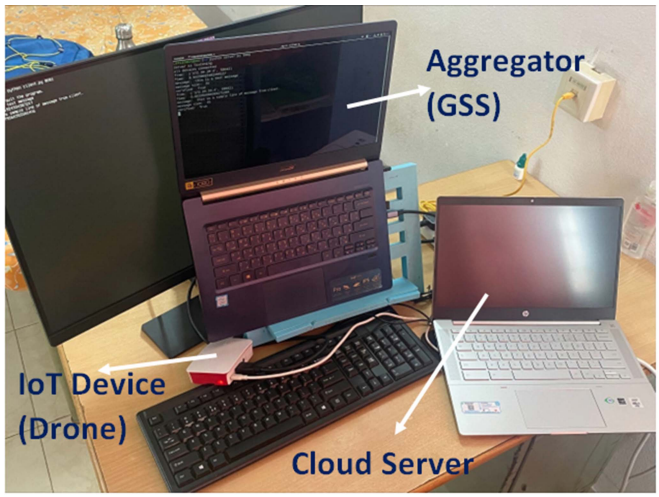


Fig. 4. Testbed setup.

In a quantum MiTM attack, the adversary blocks "all the calibration signals and transmits the forged calibration signals to disturb the activation timing calibration of the detectors". The proposed lattice-based aggregate signature scheme, uses the lattice-based quantum keys along with the blockchain technology for IoT-enabled drones applications. This will helps to achieve security against various attacks from both the classical as well as quantum computers including the "hybrid lattice-reduction," "generic" and quantum MiTM attacks.

VII. Testbed Experiments and Results

In this section, we describe the testbed experiment that was conducted to implement the proposed lattice-based aggregated signature scheme (LBAS).

A. Experimental Setup

The setup of the testbed is given in Fig. 4. The setup consists of a Raspberry Pi 3 Model B as a drone (IoT smart device), an Ubuntu 20.04 laptop as both the KGC and aggregator, and HP Chromebook laptop as a cloud server. We use the python 3.8.10 language to code the proposed LBAS. The conducted experiment confirms the message signing and verification process of our lattice based scheme for both a single message and an aggregate message. For this purpose, we have coded the $KeyGen$ algorithm in Section IV-A, the $LSSign$ signing algorithm in Section IV-B, the $LSVer$ verification algorithm in Section IV-C, the $LASign$ aggregate signing algorithm in Section IV-D and the $LAVer$ aggregate verification algorithm in Section IV-E.

The real-world scenario is simulated by creating multiple signers (users) on the Raspberry PI based IoT smart device

TABLE II
SYSTEM PARAMETERS USED IN EXPERIMENTS

Parameter/Device	Description
Drone	Raspberry Pi 3 Model B
key generation center (KGC)	Ubuntu 20.04 laptop
Aggregator (Agg)	Ubuntu 20.04 laptop
Cloud server	HP Chromebook laptop
Programming language	Python 3.8.10
p	$2^{160}+21505$
v	9
f	2^9
n (number of IoT devices/drones)	4

```
prithwi@boss:~/LatticeCrypto$ python client.py 8080
Public parameter k:  1384338
Type "exit" to quit the program.

>>>  Lattice Cryptography is fun!
Signing Time:  0.024289608001708984 s
>>>
```

Fig. 5. Experimental results at the drone 1 (IoT device) side.

```
basudeb@boss:~/LatticeCrypto$ python client.py 8080
Public parameter k:  1777715
Type "exit" to quit the program.

>>>  Hello, this is a text message.
Signing Time:  0.025076627731323242 s
>>>
```

Fig. 6. Experimental results at the drone 2 (IoT device) side.

(drone), each sending messages to the aggregator. The private and public keys for each of the signers is sent via a secure channel, while the messages and signatures are transferred over an open network. The time taken for each of the steps is measured independently.

B. Experimental System Parameters

In Table II, we describe the parameters used in our testbed experiments.

C. Results and Discussions

The messages sent by an IoT device considering as a drone are verified correctly only if the signature sent is constructed using the $LSSign$ algorithm. Also, it is hard to find different messages for which the same signature verified correctly. In Fig. 5, we have shown how drone 1 (IoT device) has created the lattice-based signature on a message with the message signing time. Similarly, we have shown how another drone, drone 2 (IoT device) has created the lattice-based signature on a message with the message signing time, which is demonstrated in Fig. 6. Now, in Fig. 7, we have shown the aggregator's individual lattice-based signature verification time for both the signed messages received from drone 1 and drone 2, respectively. In addition, we have shown the lattice-based aggregate signature generation and verification time on the received individual signed messages from the drones as well.

We have plotted the various experimental results in Fig. 8. Fig. 8(a) shows a single message signature generation cost at a drone (IoT device), whereas Fig. 8(b) tells a single message

```
(venv) LatticeCrypto:)=> python server.py 8080
Public parameter k:  1141693
Server is listening
all devices connected

From:  ('127.0.0.1', 35886)
Verification Time:  0.04947662353515625 s
Message:  Hello, this is a text message.
Verified:  True

From:  ('127.0.0.1', 35880)
Verification Time:  0.04101300239562988 s
Message:  Lattice Cryptography is fun!
Verified:  True

Messages:
Hello, this is a text message.
Lattice Cryptography is fun!
Chunk size:  24634
Aggregate Signing Time:  0.06356525421142578 s
Aggregate Verification Time:  0.06717443466186523 s
```

Fig. 7. Experimental results at the aggregator (GSS) side.

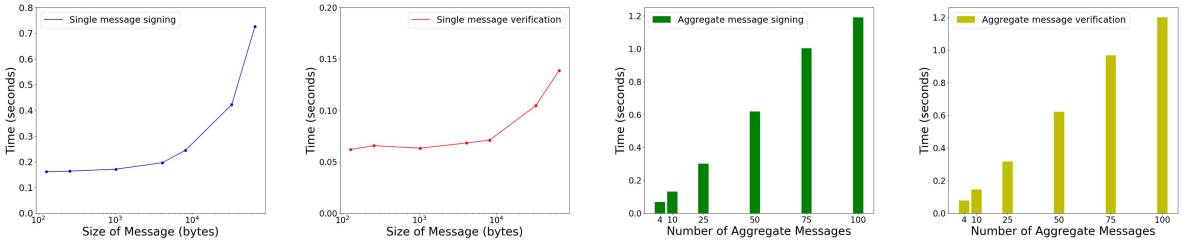
signature verification cost at a drone (IoT device). It can be observed from these figures that the time required for signing and verification of single message varies linearly with the size of messages to be signed. In Fig. 8(c), we have shown an aggregate message signature generation cost at an aggregator node by varying the number of messages to be aggregated, whereas in Fig. 8(d) we have shown an aggregate message signature verification cost at the aggregator side by varying the number of messages to be aggregated. In can be also observed from both the cases that when the number of messages is more to produce the aggregate signature, its signing cost as well as verification cost also increases linearly.

In Fig. 9(a), we have shown the histogram of times taken by $LSSign$ algorithm using 1000 random messages of size 5 KB. It can be observed from the plot that a major fraction of messages take time between 200 to 220 milliseconds. Fig. 9(b) is a similar histogram plot for the time taken by the $LSVer$ algorithm. It can be observed that the time taken lies between 60 to 70 milliseconds. Fig. 9(c) and (d) are the histogram plots for $LSSign$ and $LSVer$, respectively, using 20 KB messages. Majority of the message signing time lies between 360 to 410 milliseconds, while majority verification time lies between 80 to 110 milliseconds. We can see that as the message size increases, the variance of $LSSign$ and $LSVer$ times increases. However, this variance is small as compared to the mean values, which indicates a high reliability and robustness of our scheme.

VIII. BLOCKCHAIN SIMULATION AND RESULTS

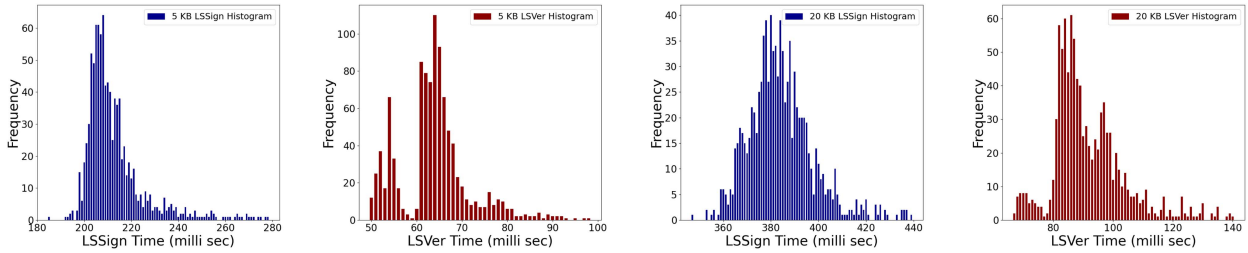
The blockchain simulations are provided in this section. For the simulation purpose, we have virtually created a "decentralized Peer-to-Peer (P2P) distributed system," where each peer or node in the network is considered as a server. Here, the total number of distributed servers is taken as 7, and the system configuration is considered as: "Ubuntu 20.04.4 LTS, Intel Core i5-4210 U CPU @ 1.70 GHz× 4, Memory 7.7 GiB, NVD7/Intel HD Graphics 4400 (HSW GT2), OS type 64-bit, disk capacity 1.0 TB".

The source code was implemented through the utilization of node.js VS code 2019 [28]. Since the blockchain is a distributed technology, adding a block into the chain requires a distributed



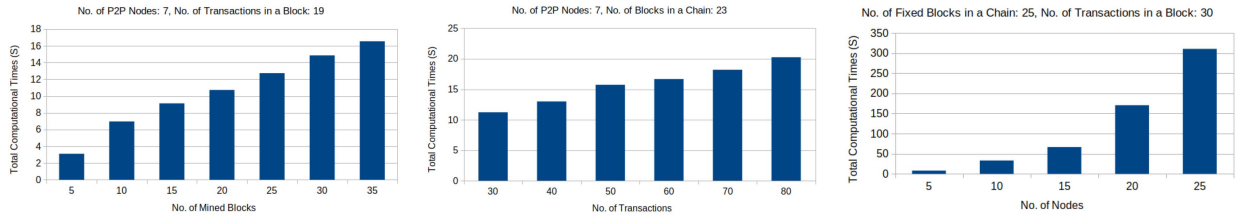
(a) (b) (c) (d)

Fig. 8. (a) Single message signature generation cost at drone (IoT device). (b) Single message signature verification cost at aggregator (GSS). (c) Aggregate signature generation cost at aggregator (GSS). (d) Aggregate signature verification cost at aggregator (GSS).



(a) (b) (c) (d)

Fig. 9. Histograms of (a) LSSign times for 5 KB message using 1000 messages. (b) LSVer times for 5 KB message. (c) LSSign times for 20 KB message. (d) LSVer times for 20 KB message.



(a) (b) (c)

Fig. 10. Blockchain simulation outcomes for (a) Case 1. (b) Case 2. (c) Case 3.

consensus mechanism. Hence, we have utilized a voting based PBFT consensus algorithm for the block mining purpose as mentioned in Section V-D. The entire blockchain simulation is performed under the following three cases:

- *Case 1:* In this context, each block is having a number of transactions which is equal to 19, and the generated blockchain comprises of a variable number of the blocks, that is, varying blockchain sizes. The simulation results shown in Fig. 10(a) indicate the computing time (in seconds), which demonstrates that the overall time for generating a blockchain with different blocks for a fixed number of transactions present in each block. Note that we have considered the synthetic data (transactions) containing in the blocks to validate the blockchain time. It has been found that as the number of blocks are increased for mining in the blockchain, the computing time also grows slowly.
- *Case 2:* In this case study, the considered blockchain has a fixed number of blocks, which is 23, where each block can carry a varying number of transactions. The simulation results displayed in Fig. 10(b) demonstrate that the computing time (the amount of time needed to build a full blockchain) increases slowly when the number of transactions considered in each block also increases.
- *Case 3:* In this scenario, we consider the number of nodes in the P2P network to be variable between 5 and 25, whereas the number of blocks in a chain or the blockchain size is constant across all the simulations. The number of transactions in a block is limited to 30, and the blockchain is also limited to 25 in size. Fig. 10(c) depicts the simulation results. According to the findings, when the number of nodes in the P2P network grows, the total computational time (in seconds) also increases.

IX. Related Work and Comparative Study

In this section, we first discuss the relevant existing basic schemes proposed in the literature. We also discuss other existing approaches that are applicable for IoT and "Internet of Drones (IoD)" applications. Next, we provide a detailed comparative

study among our proposed scheme and existing state of art schemes.

A. Existing Basic Schemes

Ma and Jiang [29] proposed a practical "lattice-based multisignature scheme using the blockchain technology" which is based a condition that the signature size is small. After that, they extended their proposed scheme to support the public key aggregation in the setting of a small signer group. In their proposed scheme, all the signers agree on selecting a random polynomial, each signer generates their own private and public keys pair by executing the key generation algorithm. Each signer runs the multisignature generation algorithm in order to generate their signature. One of the signers, called the designed combiner, who is responsible for collection of all the signatures from the signers and then computes the multisignature on the collected signatures. The verifier runs the multisignature verification algorithm for verifying the candidate multisignature. Their scheme can be also extended to support the public key aggregation through the technique described in [30], where the security of their scheme is based on the lattice-based Ring-SIS hard assumption.

Lu et al. [4] designed a "lattice-based unordered aggregate signature scheme based on the intersection method". In their approach, the unordered aggregate problem of lattice signatures with different public keys has been targeted to solve. A trusted authority, known as the KGC, who runs the setup algorithm and generates the relevant public parameters. After that, the KGC needs to execute the "key extract algorithm" for generating the public key for verification purpose, and also the secret key for each signer for signing the messages. Next, each signer needs to execute the "sign algorithm" to produce a single signature on the intended message, and the aggregator then executes the "verification algorithm" for checking all the individual signatures. Once the individual signatures are valid, the "aggregate algorithm" is executed by the aggregator for producing the aggregate signature. Finally, the aggregate signature needs to be verified by the verifier. The security of this scheme is dependent on the lattice-based "small integer solution (SIS) problem".

Shim [3] proposed an "identity-based aggregate signature (IBAS)" scheme with constant pairing computations. However, their scheme does not achieve the compactness property. The security of their scheme relies on the hardness of number theoretic "computational Diffie-Hellman (CDH) assumption". Bansarkhani et al. [33] proposed a "lattice-based sequential aggregate signature scheme" which was shown to be secure in the "random oracle model". They introduced the concept of "preimage sampleable trapdoor functions" and provided a sketch of the signature scheme.

Zhang et al. [31] proposed an efficient "homomorphic aggregate signature scheme (HASS) based on lattice". They used the idea of the "lattice-based linearly homomorphic signature scheme over a binary field in the single-user case," and then developed it into a "lattice-based HASS for the multiuser case". They proved that the security of their schemes can be achieved by reduction to the single-user case where the "signature length remains invariant".

Li et al. [32] designed a "quantum secure and non-interactive identity-based aggregate signature scheme" from the lattices. In their approach, any verifier only requires to calculate a "simple polynomial multiplication" in order to check the aggregate signature validity, because the aggregate signature size in their scheme becomes a "logarithmic function of signatures being aggregated".

Hwang and Lee [37] suggested a lightweight signature scheme which applies a "certificate-based aggregate signature". Their scheme is able to generate and verify the signed messages from the deployed IoT smart devices in an IoT-based cloud environment. Their scheme supports the key insulation property when the secret keys can be exposed due to physical attacks, like side channels attacks.

B. Existing Schemes for IoT and IoD Applications

Qian et al. [36] proposed a "lattice-based data aggregation scheme in residential networks" for IoT-enabled smart grid environment. They proposed a new "homomorphic aggregated signature based on batch RSA public key cryptosystem". They also designed a new "data aggregation scheme for the smart grid" which requires the "shorter public key" and "stronger data integrity" with respect to the traditional schemes. In addition, they utilized the "directed spanning tree based on an undirected complete graph" in order to assure the security of data aggregation structure at the base station. In their approach, the "data aggregation structure" helps in resisting attacks against outside adversaries.

Khan et al. [9] suggested a "certificate-based aggregate signature (CBS-AS) scheme based on hyperelliptic curve cryptography" in an Internet of Drones (IoD) environment. In their approach, a drone, acting as an aggregator, in a cluster can aggregate the "individual signatures of its member drones" and then verify the aggregated data. Since the drones are usually resource limited, the data aggregation by an aggregator drone may pose a limitation in this scheme.

Lu et al. [34] applied the "intersection method from lattice" in order to design a generic approach for the batch signature in an IoT network. They combined their generic approach with the "hash-and-sign paradigm" and "Fiat–Shamir transformation" for designing the batch signature methods for an IoT-based "wireless body sensor network". It was shown that their scheme is secure under the "existential unforgeability against adaptive chosen message attacks" based on the hard lattice-based problem, known as "small integer solution (SIS)" problem.

Jan and Khan [35] designed two security frameworks. The first one is on the identity-based authentication scheme, whereas the second one is an "aggregate signature-based authentication scheme" in IoD networks. However, their schemes are computationally heavy due to expensive operations like bilinear pairings.

Another "certificateless aggregate arbitrated signature" mechanism for an IoT environment was suggested by Lee et al. [38]. In their proposal, the IoT devices being the signers generate the signatures on the sensing information (messages) and the messages-cum-signatures are aggregated by the gateway node, and these are then stored at the cloud storage. A consumer being a

TABLE III
COMPARISON OF THE PROPOSED LBAS WITH BASIC LATTICE-BASED AGGREGATE SIGNATURE SCHEMES

Scheme	Public key size (in bits)	Private key size (in bits)	Single signature size (in bits)	Aggregate signature size (in bits)
Zhang *et al.* [31]	$\lambda f \log_2 p$	$f^2 \log_2 p$	$f \log_2 p$	$nf \log_2 p$
Lu *et al.* [4]	$\lambda f \log_2 p$	$f^2 \log_2 p$	$f \log_2 p$	$f \log_2 p$
Li *et al.* [32]	$f \log_2 p$	$2f \log_2 p$	$f \log_2 p$	$f \log_2 p$
Bansarkhani and Buchmann [33]	$\psi \lambda \log_2 p$	$\lambda^2 \log_2 p$	$\psi \log_2(s\sqrt{\psi})$	$\psi \log_2(s\sqrt{\psi})$
Proposed (LBAS)	$f \log_2 p$	$2f \log_2 3 \approx 3.17f$	$2f \log_2(4k_i - 63) + 1.59f$	$2f \log_2(4k_{ID_{Agg}} - 63) + 1.59f$

TABLE IV
PERFORMANCE COMPARISON WITH RELATED SCHEMES FOR IOT/IOD APPLICATIONS

Scheme	Architecture	Security	Efficiency	Security Assumption	Application Domain
Khan *et al.* [9]	Centralized	Moderate	Low	Number theoretic "elliptic curve discrete logarithm problem (ECDLP)"	"Internet of Drones (IoD)"
Lu *et al.* [34]	Centralized	High	Moderate	Lattice-based "small integer solution (SIS)" problem	"Internet of Things (IoT)"
Jan and Khan [35]	Centralized	Moderate	Low	Number theoretic "computational Diffie-Hellman problem (CDHP)"	"IoD deployment military drones"
Qian *et al.* [36]	Centralized	Moderate	Low	Number theoretic computational "integer factorization problem (IFP)" and "lattice-based homomorphic cryptosystem"	"IoT-based smart grid"
Proposed (LBAS)	Decentralized	High	High	Lattice-based Ring-LWE problem	"IoT-enabled drones"

verifier can retrieve the data and test its authenticity by verifying the signature attached in the requested message. In their scheme, the aggregate signature is generated on received messages and signatures from the deployed IoT devices. Next, the arbitrated signature of the aggregator is aggregated in "aggregate signature of IoT devices".

An aggregate signature mechanism is proposed by Gu et al. [39]. Their scheme relies on a "linearly homomorphic signature for electronic healthcare systems (EHS) in an IoT-based helathcare system". It supports both the aggregation and linear homomorphism properties, and also uses the double data compression. Thumbur et al. [40] also suggested a "certificateless aggregate signature-based authentication scheme" for an IoT-enabled vehicular ad hoc network (VANET) environment. Since their scheme is not based on the expensive pairing operations, the computational efficiency of their proposed system is improved.

C. Comparative Study

We now explain a comparison study among our proposed scheme (LBAS) and other existing basic lattice-based aggregate signature schemes, namely the schemes of Zhang et al. [31], Lu et al. [4], Li et al. [32], and Bansarkhani and Buchmann [33]. Table III shows that the public key size of our scheme is less than the public key size of the schemes [4], [31]. Similarly, the size of the private key in our scheme is also small with respect to that for the schemes [4], [31], [32]. On the other hand, our scheme requires small individual signature size as well as aggregate signature size with respect to all other schemes. Note that $\psi \geq 5\lambda \log_2 p$ is an integer, f is defining as a power of 2 where f is the highest power of an irreducible polynomial, p is a large prime, n is the total number of signers, λ is a security parameter, k_i and $k_{ID_{Agg}}$ are the positive integers less than $\sqrt{p}$, respectively, and s is the "Gaussian parameter". Since the proposed scheme relies on the hardness of the lattice-based "Ring-LWE" problem, it resists various quantum attacks as explained in Section VI-C5.

In Table IV, we have also compared the proposed LBAS with the existing related schemes for IoT/IoD applications, like the schemes of Khan et al. [9], Lu et al. [34], Jan and Khan [35], and Qian et al. [36]. It is noted that the proposed LBAS only supports decntralized architecture as it is based on the blockchain technology, whereas other schemes are centralized in nature. Moreover, the proposed LBAS is very efficient and also provide high-level security including the various quantum attacks (explained in Section VI-C5) as compared to the schemes [9], [35], [36].

X. CONCLUSION

In this article, we designed a lattice-based aggregate signature scheme whose security is based on the hardness of the lattice-based Ring-LWE problem. We then applied the designed lattice-based aggregate signature scheme for the real-time IoD applications, where the IoT-enabled drones play the role of individual signers and the GSS plays the role of an aggregator. Next, the aggregate message with its aggregate signature is verified by a cloud server in the P2P cloud servers network and the blocks containing the verified transactions are formed by the cloud server for mining purpose and the created blocks are then added into the blockchain network for proving immutability, decentralization and transparency purposes. The conducted formal and informal security analysis exhibited the strong security of the proposed scheme against a variety of attacks including quantum attacks. A real test-bed experiment was conducted to measure various computational time needed for individual signature generation and verification, and aggregate signature generation and verification by the respective individual signer being an IoT device (acting as a drone), the GSS and the cloud server, respectively. A blockchain based simulation was carried out to show the effect on computational time. Finally, a comparative analysis of the proposed scheme shows superior security and efficiency as compared to those for other competing relevant schemes.

In future, we would like to fine-tune the proposed scheme to convert it into a more lightweight scheme. It will be helpful in reducing the computational costs for signing and verifying the lattice-based aggregate signatures. Since the aggregate signatures are used inside the blocks which are mined and added into the blockchain via a consensus process, so the more lightweight aggregate signature generation and verification will help in reducing the blocks verification and addition time during the mining process too.

ACKNOWLEDGMENT

The authors would like to thank the Associate Editor and reviewers who helped us to enrich the quality of the manuscript.

REFERENCES

[1] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM J. Comput.*, vol. 26, pp. 1484–1509, 1997.

[2] P. Bert, P.-A. Fouque, A. Roux-Langlois, and M. Sabt, "Practical implementation of Ring-SIS/LWE based signature and IBE," in *Proc. Int. Conf. Post-Quantum Cryptogr.*, 2018, pp. 271–291.

[3] K.-A. Shim, "An ID-based aggregate signature scheme with constant pairing computations," *J. Syst. Softw.*, vol. 83, pp. 1873–1880, 2010.

[4] X. Lu, W. Yin, Q. Wen, Z. Jin, and W. Li, "A lattice-based unordered aggregate signature scheme based on the intersection method," *IEEE Access*, vol. 6, pp. 33986–33994, 2018.

[5] A. Kumar, D. Augusto de Jesus Pacheco, K. Kaushik, and J. J. Rodrigues, "Futuristic view of the internet of quantum drones: Review, challenges and research agenda," *Veh. Commun.*, vol. 36, 2022, Art. no. 100487.

[6] C. C. Baseca, J. R. Diaz, and J. Lloret, "Communication ad hoc protocol for intelligent video sensing using AR drones," in *Proc. IEEE 9th Int. Conf. Mobile Ad-hoc Sensor Netw.*, 2013, pp. 449–453.

[7] L. Wan, G. Han, L. Shu, N. Feng, C. Zhu, and J. Lloret, "Distributed parameter estimation for mobile wireless sensor network based on cloud computing in battlefield surveillance system," *IEEE Access*, vol. 3, pp. 1729–1739, 2015.

[8] I. Garcia-Magarino, R. Lacuesta, M. Rajarajan, and J. Lloret, "Security in networks of unmanned aerial vehicles for surveillance with an agent-based approach inspired by the principles of blockchain," *Ad Hoc Netw.*, vol. 86, pp. 72–82, 2019.

[9] M. A. Khan, I. Ullah, M. H. Alsharif, A. H. Alghtani, A. A. Aly, and C.-M. Chen, "An efficient certificate-based aggregate signature scheme for Internet of Drones," *Secur. Commun. Netw.*, vol. 2022, 2022, Art. no. 9718580.

[10] L. D. P. Pugliese, F. Guerriero, and G. Macrina, "Using drones for parcels delivery process," *Procedia Manuf.*, vol. 42, pp. 488–497, 2020.

[11] R. H. Kabir and K. Lee, "Wildlife monitoring using a Multi-UAV system with optimal transport theory," *Appl. Sci.*, vol. 11, no. 9, pp. 1–22, 2021.

[12] B. Ivosevic, Y.-G. Han, Y. Cho, and O. Kwon, "The use of conservation drones in ecology and wildlife research," *J. Ecol. Environ.*, vol. 38, no. 1, pp. 113–118, 2015.

[13] A. Hafeez et al., "Implementation of drone technology for farm monitoring & pesticide spraying: A review," *Inf. Process. Agriculture*, 2022. [Online]. Available: https://doi.org/10.1016/j.inpa.2022.02.002

[14] B. Narwal and A. K. Mohapatra, "A survey on security and authentication in wireless body area networks," *J. Syst. Architecture*, vol. 113, 2021, Art. no. 101883.

[15] D. Dolev and A. Yao, "On the security of public key protocols," *IEEE Trans. Inf. Theory*, vol. 29, no. 2, pp. 198–208, Mar. 1983.

[16] R. Canetti and H. Krawczyk, "Universally composable notions of key exchange and secure channels," in *Proc. Int. Conf. Theory Appl. Cryptographic Techn.*, 2002, pp. 337–351.

[17] T. S. Messerges, E. A. Dabbish, and R. H. Sloan, "Examining smart-card security under the threat of power analysis attacks," *IEEE Trans. Comput.*, vol. 51, no. 5, pp. 541–552, May 2002.

[18] J. N. Ortiz, R. R. de Araujo, D. F. Aranha, S. I. R. Costa, and R. Dahab, "The Ring-LWE problem in lattice-based cryptography: The case of twisted embeddings," *Entropy*, vol. 23, no. 9, pp. 1–23, 2021.

[19] J. Hoffstein, N. Howgrave-Graham, J. Pipher, and W. Whyte, "Practical Lattice-Based Cryptography: NTRUEncrypt and NTRUSign," in *The LLL Algorithm.* Berlin, Germany: Springer, 2010, pp. 349–390.

[20] T. Guneysu, V. Lyubashevsky, and T. Poppelmann, "Practical lattice-based cryptography: A signature scheme for embedded systems," in *Cryptographic Hardware and Embedded Systems–CHES*, Berlin, Germany: Springer, 2012, pp. 530–547.

[21] V. Lyubashevsky, "Lattice signatures without trapdoors," in *Proc. 31st Annu. Int. Conf. Theory Appl. Cryptographic Techn.*, D. Pointcheval and T. Johansson, Eds. Cambridge, U.K., 2012, pp. 738–755.

[22] A. K. Das, B. Bera, M. Wazid, S. S. Jamal, and Y. Park, "iGCACS-IoD: An improved certificate-enabled generic access control scheme for internet of drones deployment," *IEEE Access*, vol. 9, pp. 87024–87048, 2021.

[23] B. Bera, A. Vangala, A. K. Das, P. Lorenz, and M. K. Khan, "Private blockchain-envisioned drones-assisted authentication scheme in IoT-enabled agricultural environment," *Comput. Standards Interfaces*, vol. 80, 2022, Art. no. 103567.

[24] S. Yu, A. K. Das, Y. Park, and P. Lorenz, "SLAP-IoD: Secure and lightweight authentication protocol using physical unclonable functions for internet of drones in smart city environments," *IEEE Trans. Veh. Technol.*, vol. 71, no. 10, pp. 10374–10388, Oct. 2022.

[25] H. Zhang, J. Wang, and Y. Ding, "Blockchain-based decentralized and secure keyless signature scheme for smart grid," *Energy*, vol. 180, pp. 955–967, 2019.

[26] M. Castro and B. Liskov, "Practical byzantine fault tolerance and proactive recovery," *ACM Trans. Comput. Syst.*, vol. 20, no. 4, pp. 398–461, 2002.

[27] M. Bellare and G. Neven, "Multi-signatures in the plain public-key model and a general forking lemma," in *Proc. 13th ACM Conf. Comput. Commun. Secur.*, 2006, pp. 390–399.

[28] K. Khullar, "Implementing PBFT in blockchain," 2019. Accessed: Mar. 2022. [Online]. Available: https://medium.com/coinmonks/implementing-pbft-in-blockchain-12368c6c9548

[29] C. Ma and M. Jiang, "Practical lattice-based multisignature schemes for blockchains," *IEEE Access*, vol. 7, pp. 179765–179778, 2019.

[30] G. Maxwell, A. Poelstra, Y. Seurin, and P. Wuille, "Simple schnorr multi-signatures with applications to bitcoin," *Des., Codes Cryptogr.*, vol. 87, pp. 2139–2164, 2019.

[31] P. Zhang, J. Yu, and T. Wang, "An efficient homomorphic aggregate signature scheme based on lattice," *Math. Problems Eng.*, vol. 2014, 2014, Art. no. 9.

[32] Q. Li, M. Luo, C. Hsu, L. Wang, and D. He, "A quantum secure and noninteractive identity-based aggregate signature protocol from lattices," *IEEE Syst. J.*, vol. 16, no. 3, pp. 4816–4826, Sep. 2022.

[33] R. El Bansarkhani and J. Buchmann, "Towards lattice based aggregate signatures," in *Proc. Int. Conf. Cryptol.*, 2014, pp. 336–355.

[34] X. Lu, W. Yin, Q. Wen, K. Liang, L. Chen, and J. Chen, "Message integration authentication in the Internet-of-Things via lattice-based batch signatures," *Sensors*, vol. 18, no. 11, 2018, Art. no. 4056.

[35] S. U. Jan and H. U. Khan, "Identity and aggregate signature-based authentication protocol for IoD deployment military drone," *IEEE Access*, vol. 9, pp. 130247–130263, 2021.

[36] J. Qian, Z. Cao, M. Lu, X. Chen, J. Shen, and J. Liu, "The secure lattice-based data aggregation scheme in residential networks for smart grid," *IEEE Internet Things J.*, vol. 9, no. 3, pp. 2153–2164, Feb. 2022.

[37] Y.-W. Hwang and I.-Y. Lee, "A lightweight certificate-based aggregate signature scheme providing key insulation," *Comput., Mater. Continua*, vol. 69, no. 2, pp. 1747–1764, 2021.

[38] D.-H. Lee, K. Yim, and I.-Y. Lee, "A certificateless aggregate arbitrated signature scheme for IoT environments," *Sensors*, vol. 20, no. 14, 2020, Art. no. 3983.

[39] Y. Gu, L. Shen, F. Zhang, and J. Xiong, "Provably secure linearly homomorphic aggregate signature scheme for electronic healthcare system," *Mathematics*, vol. 10, no. 15, 2022, Art. no. 2588.

[40] G. Thumbur, G. S. Rao, P. V. Reddy, N. B. Gayathri, D. V. R. K. Reddy, and M. Padmavathamma, "Efficient and secure certificateless aggregate signature-based authentication scheme for vehicular Ad Hoc networks," *IEEE Internet Things J.*, vol. 8, no. 3, pp. 1908–1920, Feb. 2021.