Quantum Safe Lattice-Based Single Round Online Collaborative Multi-Signature Scheme for Blockchain-Enabled IoT Applications

PRITHWI BAGCHI, International Institute of Information Technology, Hyderabad, India
BASUDEB BERA, National University of Singapore, Singapore
ASHOK KUMAR DAS*, International Institute of Information Technology, Hyderabad, India and Korea University, South Korea (*Corresponding author)
BIPLAB SIKDAR, National University of Singapore, Singapore

Multi-signature protocols allow a group of signers to collectively generate a single signature for a shared message. In the context of a decentralized blockchain, multi-signature schemes play a pivotal role in reducing the signature size. Recently, several multi-signature methods have emerged in the literature, some operating in discrete-log settings and others in lattice settings. However, many of the existing lattice-based multi-signature schemes incur high computation costs and online round complexity. Traditional public key-based multi-signature schemes are susceptible to quantum threats and they are computationally intensive as well. A lattice-based multi-signature can provide robust security, which often falls short in terms of efficiency when it comes to round complexity. In this article, we aim to introduce a single-round lattice-based multi-signature scheme specifically designed for decentralized public blockchains. What sets the proposed scheme apart is its ability to function without the need for trapdoor commitments or sample pre-images, which are common features in existing lattice-based signature methods. Furthermore, we explore some potential applications in a generic Internet of Things (IoT) environment and their integration of the proposed scheme with the blockchain technology. The security of the proposed scheme is based on lattice-hard problems, like Ring-SIS (Shortest Integer Solution) and Ring-LWE (Learning With Errors).

CCS Concepts: • **Security and privacy → Security protocols**; **Authentication**.

Additional Key Words and Phrases: Post-quantum security, Lattice, Internet of Things (IoT), Collaborative multi-signature, Blockchain

ACM Reference Format:
Prithwi Bagchi, Basudeb Bera, Ashok Kumar Das*, and Biplab Sikdar. 2024. Quantum Safe Lattice-Based Single Round Online Collaborative Multi-Signature Scheme for Blockchain-Enabled IoT Applications. *ACM Trans. Sensor Netw.* xx, 4, Article xxx (September 2024), 32 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Internet of Things (IoT) refers to a network of physical devices embedded with sensors, software, and connectivity capabilities that enable them to collect and exchange data over the Internet. These devices, also known as "smart"

Authors' addresses: Prithwi Bagchi, International Institute of Information Technology, Hyderabad, Center for Security, Theory and Algorithmic Research, Hyderabad, India, prithwi.bagchi@research.iiit.ac.in; Basudeb Bera, National University of Singapore, Department of Electrical and Computer Engineering, Singapore, b.bera26@nus.edu.sg; Ashok Kumar Das*, International Institute of Information Technology, Hyderabad, Center for Security, Theory and Algorithmic Research, Hyderabad, India and Korea University, Department of Computer Science and Engineering, College of Informatics, Seoul, South Korea (*Corresponding author), iitkgp.akdas@gmail.com,ashok.das@iiit.ac.in; Biplab Sikdar, National University of Singapore, Department of Electrical and Computer Engineering, Singapore, bsikdar@nus.edu.sg.

ACM 1550-4859/2024/9-ARTxxx
https://doi.org/XXXXXXX.XXXXXXX

or “connected" devices, can encompass a wide range of items, from everyday objects, such as home appliances and wearables, to industrial machinery and infrastructure components [1], [24]. By connecting devices to the Internet, IoT enables the collection, analysis, and remote control or monitoring of these devices.

There are numerous real-life applications of IoT across various industries. One such application is the smart home [53], [58], where IoT empowers homeowners to control and automate various aspects of their homes, including lighting, temperature, security systems, and appliances. Another notable application is Industrial IoT [34], [39], where IoT plays a vital role in industrial settings, facilitating automation, monitoring, and process optimization. Additionally, IoT is extensively utilized in the creation of smart cities [15], [27], [45], healthcare systems [50], [60], smart grid [54] and so on. The potential of IoT to impact numerous other sectors, including transportation, insurance, and hospitality, is also noteworthy. The widespread adoption of IoT is driving innovation and transforming various aspects of our daily lives and industries. Most of the IoT devices are typically connected to the insecure (public) Internet or wireless media, which exposes them to potential cyber threats and attacks. These attacks can target the devices themselves as well as the data they collect and transmit as the data is confidential and private in many cases. Thus, if some proper security measures are not in place, the data can be intercepted, manipulated, or even stolen, leading to privacy breaches and identity theft [52], [56], [57]. Most IoT applications currently rely on traditional public-key based cryptographic algorithms, like Rivest-Shamir-Adleman (RSA), Elliptic Curve Cryptography (ECC), and digital signatures. However, these algorithms are susceptible to attacks from quantum computers, which possess superior computational power. Therefore, implementing post-quantum cryptography in IoT systems is essential to withstand such attacks and provide future-proof security, data protection, secure key exchange protocols, and device authentication [59].

A multi-signature is a cryptographic technique, where a group of signers having their own public-private keys pair, jointly generate a multi-signature on a common message. Multi-signature in IoT applications helps in enhancing security, preventing unauthorized actions, ensuring distributed control, and establishing accountability. By requiring multiple trusted signatures, it adds an extra layer of protection and reliability in IoT systems. Based on the literature survey, various multi-signature schemes have been developed based on traditional public-key cryptographic systems, such as modular exponentiation (RSA), discrete logarithm problem (DLP), or bilinear-pairing assumption. However, with an exponential growth of quantum cryptosystems, these existing public-key based cryptosystems may face significant threats in near future. To counter these threats, the adoption of post-quantum cryptographic techniques based on lattice structures have become widespread. While some lattice-based multi-signature schemes have been proposed, such as the schemes presented by Ma and Jiang [41], Boschini *et al.* [11], Fleischhacker *et al.* [22], Kansal *et al.* [31], and Bansarkhani and Sturm [21], but none of these schemes have been validated or supported for real-life IoT applications.

The development of a multi-signature scheme faces several significant technical obstacles, including: a) achieving security in the plain public-key (PPK) model, b) achieving concurrent security, c) reducing the online round complexity, and d) achieving the key-aggregation technique. To support the PPK model, it is necessary for each signer to publish their public key without engaging in any interactive key-generation algorithm. Additionally, an adversary should not be able to convince the verifier that an honest party, denoted by PA, participated in the signing algorithm of the multi-signature scheme unless the PA explicitly agreed to it. While existing multi-signature schemes, such as those presented in [5], [6], [42], [48], [57] have provable security features against concurrent attacks, they typically involve a two-round online phase in the signing algorithm. Even the scheme proposed in [41] requires a four-round online phase, which results in high online round complexity for participants. To address these issues and reduce online round complexity, it is preferable in practice to pre-process the computation of the interaction process without knowledge of the message being signed. This offline-online method is commonly employed in the context of multi-party computation [17]. Recently, the authors in [33] and [47] introduced Schnorr-based multi-signatures that offer a remarkable feature: “they require only a single round of interaction within the online phase while maintaining security against concurrent attacks”. These schemes

also utilize a backward key aggregation technique, which is a significant optimization strategy that combines a set of public keys into a single aggregated Schnorr public key. Backward key aggregation holds great significance for a designed multi-signature scheme as it allows the verifiers to validate a signature using a standard Schnorr public key, by enabling interoperability with the existing verification algorithms.

1.1 Motivation and Contributions

The majority of existing multi-signature schemes rely on conventional public key-based cryptosystems, while only a small fraction of them are based on lattice-based cryptosystems. However, it is noteworthy that most lattice-based multi-signatures exhibit significant online round complexity. As a result, these schemes may not be considered practical for IoT applications due to their lack of relevance in the IoT domain. To address this issues and strengthen resistance against quantum attacks, we introduce a novel post-quantum lattice-based single-round multi-signature scheme, specifically designed for IoT applications.

The major contributions are outlined as follows:

- We utilized widely-adopted hardness assumptions, namely Ring-SIS [7] and Ring-LWE [10], in the design of our multi-signature scheme for blockchain-based IoT applications. Additionally, we incorporated the "single round online phase" in the signing algorithm to ensure that our scheme achieves a low round complexity. The blockchain has been used as an add-on service for secure data storage purpose.
- The scheme supports each signer in pre-processing the offline phase before the "single round online phase". During this offline phase, all the signers engage in interactions and exchange a set of "commit" messages among themselves. Subsequently, each party employs a "random linear combination" approach, similar to [11], to aggregate all the "commit" messages and utilize them to generate the corresponding signature of the signer. This process enhances the security level, which also ensures the generation of a secure multi-signature. Without the participation of all signers, a multi-signature cannot be generated, thereby making it impossible for an adversary to produce a fraudulent multi-signature without knowledge of all signers' secret keys.
- The proposed scheme includes both formal and informal security analyses, where the informal security analysis confirms that the scheme is capable of withstanding various attacks, including quantum attacks, and the formal security analysis establishes the correctness (mathematically) of the scheme in the presence of quantum threats.
- A blockchain simulation has been conducted to measure the computational time required for mining blocks in the blockchain. The simulation involves varying the number of transactions and blocks in the chain, aiming to demonstrate the feasibility of the proposed scheme.
- A comparative study is also presented, featuring several existing lattice-based multi-signature schemes, to showcase the effectiveness of the proposed scheme in IoT applications. The comparison is based on various factors, including public key size, secret key size, aggregated public key size, and multi-signature size. Additionally, the costs associated with the generation and verification of multi-signatures are provided for both the proposed scheme and existing schemes, thereby highlighting the scalability of the proposed scheme as well.

1.2 Article Outline

The paper organization is as follows. The essential lattice-based mathematical preliminaries are discussed in Section 2. In Section 3, we discuss the related work associated with the lattice-based multi-signature schemes. The system model containing the network and threat models is then provided in Section 4. The basic lattice-based single round online collaborative multi-signature scheme has been presented in Section 5. Next, the designed basic collaborative multi-signature scheme has been applied in IoT applications using blockchain-as-a-service

in Section 6. The detailed security analysis with both the formal and informal security is provided in Section 7. A comparative analysis among the proposed LBCMS and other existing approaches is given in Section 8. The blockchain simulation and its various results associated with the proposed scheme are discussed in Section 9. The conclusion and some future research directions are finally provided in Section 10.

2 MATHEMATICAL PRELIMINARIES

Let $\mathbb{N}$ represent the set of all natural numbers and let α be an element, $\alpha \in \mathbb{N}$, such that $f = 2^{\alpha} \in \mathbb{N}$. Consider a large prime q that satisfies $q \equiv 1 \pmod{2f}$. Now, consider a finite field $\mathbb{Z}_q = \{0, 1, \cdots, q-1\}$ and the ideal $< x^f + 1 >$ is generated by the irreducible polynomial $x^f + 1$. We define a ring R as $R = \mathbb{Z}[x]/< x^f + 1 >$, which can be expressed as $R = \{\Psi(x) : \Psi(x) = \sum_{u=0}^{f-1} B_u x^u : \forall u \in \{0, 1, \cdots, f-1\}, B_u \in \mathbb{Z}\}$, where $\mathbb{Z}$ is the set of all integers. Similarly, we define a finite field $R_q = \mathbb{Z}_q[x]/< x^f + 1 >= \{\Psi(x) : \Psi(x) = \sum_{u=0}^{f-1} D_u x^u : \forall u \in \{0, 1, \cdots, f-1\}, D_u \in \mathbb{Z}_q\}$. Let d be defined as $0 < d < \sqrt{q}$ with the constraint that $d \leq (q-1)/2$. We define $R_{q,d}$ as a subset of R_q, that is, $R_{q,d} = \{\Psi(x) : \Psi(x) = \sum_{u=0}^{f-1} D_u x^u : \forall u \in \{0, 1, \cdots, f-1\}, D_u \in [-d, d]\}$. We introduce the following hash functions: H_1, H_2, H_3, H_4, and H_5, which are defined as $H_1 : \{0,1\}^* \rightarrow C_{32}$, $H_2 : \{0,1\}^* \rightarrow \mathbb{Z}_q - \{0\}$, where $C_\kappa = \{c \in R : ||c||_\infty = 1 \wedge ||c||_1 = \kappa\}$, $H_3 : \{0,1\}^f \rightarrow R_{q,d}$, $H_4 : R_q \rightarrow S_1$, and $H_5 : \{0,1\}^* \rightarrow C_{32}$, where $p \geq 1$, the norm l_p on a vector x is $||x||_p = \sqrt[p]{\sum_{u=0}^{f-1} |x_u|^p}$, and the sup-norm on the vectors $x_1, x_2 \cdots, x_f$ is defined as $||x||_\infty = \max\{|x_u|; u \in [f-1] \cup \{0\}\}$. Note that H_5 is different from H_1, and S_η is defined as $S_\eta = \{x \in R; ||x||_\infty \leq \eta\}$.

2.1 Lattice-based Cryptography

Let $\mathbb{R}^f$ be an f-dimensional real space. The lattice $\widehat{\mathcal{L}}$ of an f-dimensional real space is a discrete subgroup of $\mathbb{R}^f$. The lattice $\widehat{\mathcal{L}}$ is said to be a "full-rank lattice" if $\text{Span}(\widehat{\mathcal{L}}) = \mathbb{R}^f$. When the lattice $\widehat{\mathcal{L}}$ is considered as a subset of $\mathbb{Z}^f$, it is called the integer lattice. A basis is formed from a set of independent vectors that generate any point in the lattice $\widehat{\mathcal{L}}$. Thus, the basis of $\widehat{\mathcal{L}}$ is represented by a set $Basis = \{\widehat{b_1}, \widehat{b_2}, \cdots, \widehat{b_f}\}$, such that $\widehat{\mathcal{L}}\ (Basis) = \{\sum_{u=1}^{f} x_u.\widehat{b_u} | x_u \in \mathbb{Z}\}$, where each $\widehat{b_u} \in \mathbb{R}^f$, $\forall u \in [f]$, where $[f] = \{1, 2, \cdots, f\}$. The minimum distance of the lattice $\widehat{\mathcal{L}}\ (Basis)$ can be defined as $\lambda_1{}^p(\widehat{\mathcal{L}}\ (Basis)) = \min\{||x||_p; x \in \widehat{\mathcal{L}}(Basis) - \{0\}\}$.

2.2 Computational Hard Problems

In the following, we discuss the following computational hard problems related to the security of the proposed scheme in this article.

2.2.1 Learning with Errors (LWE). Let $\overline{s} \in_R \mathbb{Z}_q^f$ be a fixed vector, and χ be an error distribution over $\mathbb{Z}$. We randomly select a vector $a \in \mathbb{Z}_q^f$ from a uniform distribution over $\mathbb{Z}_q^f$ and a number $\bar{e} \in \mathbb{Z}$ referred to as an error. We then compute $t = (a.\overline{s} + \bar{e}) \pmod{q}$. Given $\varpi\ (\geq f)$ samples of the form $\{(a_i, a_i.\overline{s} + \bar{e_i} \pmod{q}); i \in [\varpi]\}$, the task is to recover the unique random secret $\overline{s}$. It is worth noting that when the secret $\overline{s}$ is sampled from the same error distribution as $\bar{e}$, the hardness of the LWE problem remains valid [2], [49].

2.2.2 Ring-Learning with Errors (Ring-LWE). This problem is another important aspect of lattice-based cryptography (LBC), and the security of many lattice-based schemes relies on the hardness of the Ring-LWE problem [10], [2]. The following is the Ring-LWE problem.

Choose a polynomial $a \in R_q$ of degree at most $f-1$ and $\overline{s} \in_R R_q$ of degree at most $f-1$. Additionally, consider an error distribution χ over the set R. Consider the Ring-LWE distribution $D_{\overline{s},\chi}$, which produces the output $(a, a.\overline{s} + e \pmod{x^f + 1}) \in R_q \times R_q$, where e is the error vector chosen uniformly at random from χ. Choose $a_i \in R_q$ and e_i uniformly at random from χ for $i \in [\varpi]$, where $\varpi \geq f$. This problem is associated with

recovering the secret $\bar{s} \in_R R_q$ with a uniform distribution over $\bar{s} \in_R R_q$ from ϖ samples of the form: $\{(a_i, a_i.\bar{s} + e_i \pmod{x^f + 1}); i \in [\varpi]\}$.

2.2.3 Ring-Short Integer Solution (Ring-SIS). A vector of ϖ polynomials $a = (a_1, a_2, \cdots, a_\varpi)^t$, chosen uniformly at random from ${R_q}^\varpi$, is provided. The task is to find an $\bar{s} \in R^\varpi$ in such a way that $a^t.\bar{s} = 0 \pmod{q}$ together with $0 < ||\bar{s}|| \leq \beta$ holds, where β is a given positive real number and a^t is the transposition of vector a.

2.3 Binary Decomposition of R_q Elements

If there is an element $a \in_R R_q$, where a can be represented as $(a_0, a_1, \cdots, a_{f-1})$, the binary decomposition of each a_u, for $u \in \{0, 1, \cdots, f-1\}$, is denoted as $(a_{u,0}, a_{u,1}, \cdots, a_{u,\lceil \log q \rceil - 1}) \in \{0,1\}^{\lceil \log q \rceil}$, that is, $a_u = \sum_{j=0}^{\lceil \log q \rceil - 1} a_{u,j}.2^j$. The binary decomposition is considered as a function $bin_q : R_q \rightarrow {R_q}^{\lceil \log q \rceil}$ such that $bin_q(a) = bin_q(a_0, a_1, \cdots, a_{f-1}) = (\sum_{u=0}^{f-1} a_{u,0}.x^u, \sum_{u=0}^{f-1} a_{u,1}.x^u, \cdots, \sum_{u=0}^{f-1} a_{u,\lceil \log q \rceil - 1}.x^u)$, for some $a \in R_q$ [22].

2.4 Projection onto R_q Elements

It is a mapping defined as $proj_q : {R_q}^{\lceil \log q \rceil} \rightarrow R_q$, such that $proj_q(b) = \sum_{j=0}^{\lceil \log q \rceil - 1} 2^j.b_j$, for some $b \in {R_q}^{\lceil \log q \rceil}$ [22].

Lemma 2.1. *For all $a = (a_0, a_1, \cdots, a_{f-1}) \in R_q$, $proj_q(bin_q(a)) = a$.*

Proof. We have,

$$\begin{aligned}
proj_q(bin_q(a)) &= proj_q(bin_q(a_0, a_1, \cdots, a_{f-1})) \\
&= proj_q(\sum_{u=0}^{f-1} a_{u,0}.x^u, \cdots, \sum_{u=0}^{f-1} a_{u,\lceil \log q \rceil - 1}.x^u) \\
&= \sum_{j=0}^{\lceil \log q \rceil - 1} 2^j (\sum_{u=0}^{f-1} a_{u,j}.x^u) \\
&= \sum_{u=0}^{f-1} (\sum_{j=0}^{\lceil \log q \rceil - 1} 2^j.a_{u,j}) x^u \\
&= \sum_{u=0}^{f-1} a_u.x^u = (a_0, a_1, \cdots, a_{f-1}) \\
&= a.
\end{aligned}$$

Hence, the result follows. □

Lemma 2.2. *$proj_q$ is a linear mapping, that is, for any two $b^0, b^1 \in {R_q}^{\lceil \log q \rceil}$ and any $\Psi_0, \Psi_1 \in R_q$, $proj_q(\Psi_0.b^0 + \Psi_1.b^1) = \Psi_0.proj_q(b^0) + \Psi_1.proj_q(b^1)$.*

Proof. It follows that

$$\begin{aligned}
proj_q(\Psi_0.b^0 + \Psi_1.b^1) &= \sum_{j=0}^{\lceil \log q \rceil - 1} 2^j.(\Psi_0.b_j^0 + \Psi_1.b_j^1) \\
&= \Psi_0.(\sum_{j=0}^{\lceil \log q \rceil - 1} 2^j.b_j^0) + \Psi_1.(\sum_{j=0}^{\lceil \log q \rceil - 1} 2^j.b_j^1) \\
&= \Psi_0.proj_q(b^0) + \Psi_1.proj_q(b^1).
\end{aligned}$$

□

2.5 Generalized Offline-Online Multi-signature

A multi-signature involves seven algorithms, namely SetUp, KeyGen, KeyAgg, SignOff, SignOn, Agg and Ver [11]. Assume that there are N signers in the multi-signature scheme. The details of these algorithms are explained below.

- **SetUp**$(1^\lambda) \rightarrow pp$
  The input is a security parameter λ, and the outputs are the public parameters pp.
- **KeyGen**$(pp) \rightarrow \{(pk_i, sk_i); i \in [N]\}$
  In this algorithm, all the signers i generate their own public keys and secret keys. Thus, for each $i \in [N]$, the i^{th} signer obtains its public key pk_i and secret key sk_i.
- **KeyAgg**$(pk_i; i \in [N]) \rightarrow AGK$
  This algorithm takes the public keys of all the signers as input and then produces the aggregated public key AGK deterministically.
- **MSign (Offline)**
  The offline signing algorithm is independent of a message ϕ to be signed. Each signer i executes this phase before signing the message ϕ. For each $i \in [N]$, the i^{th} signer generates an offline message Off_i along with a state information st.
- **MSign (Online)**
  This algorithm is also executed by each signer. For each $i \in [N]$, the i^{th} signer takes the input as the set of offline messages $\{Off_u : u \in [N]\}$, state information st generated in the offline phase of the multi-signature generation, message ϕ, secret key sk_i, and public keys of all the signers $\{pk_u : u \in [N]\}$. The output is a signature Z_i.
- **Agg**$(Z_1, \cdots, Z_N)$
  This algorithm takes the signatures of the signers as input and outputs an aggregated signature σ_{Agg}.
- **Ver**$(AGK, \phi, \sigma_{Agg})$
  The input of this algorithm includes the aggregated public key AGK, message ϕ, and aggregated signature σ_{Agg}. The output is either 0 (invalid) or 1 (valid).

3 RELATED WORK

Changshe *et al.* [41] proposed a practical lattice-based multisignature scheme (PLMS) for blockchains. Their scheme requires smaller signature size as compared to Bansarkhani *et al.*'s scheme [21]. Afterwards, they extended the PLMS scheme to the "extended lattice-based multisignature scheme (ELMS)" [41] in order to enable public key aggregation with approximately the same performance. Both the PLMS scheme and its extended version have a four-round online phase to generate the associated multi-signature, which is not efficient. In the multi-signature generation algorithm in [41], each party runs the four-round online multisignature (MSign) algorithm that requires the total complexity of 2^α corresponding to each message, where $\alpha \geq N$ and N is the number of signers in the multi-signature. Thus, this scheme has high computational complexity. It is worth noticing that the communication complexity of the third round phase in PLMS is 2^α. They demonstrated that their scheme is provably secure in the random oracle model under the Ring-SIS assumption. They selected the parameters appropriately for their multi-signature scheme and also provided experimental data.

Fleischhacker *et al.* [22] proposed a synchronized multi-signature scheme that relied on the Ring-SIS lattice assumption. The main focus of their scheme is synchronized settings, where the current time-step serves as an additional input in the signing algorithm. They ensured that a signer cannot sign multiple messages within a single time-step. Initially, Fleischhacker *et al.* [22] defined a homomorphic vector commitment scheme (HVC)

on R_q and a *label* function, and also constructed another HVC scheme on R_q^{ξ} based on a labeled binary tree and "key-homomorphic one-time signature", where ξ is a positive integer. An advantage of this scheme is that it is non-interactive.

Kansal *et al.* [31] proposed a multi-signature scheme based on a lattice-based cryptosystem, which uses the Ring-SIS assumption. The communication and storage costs of their scheme are of complexity $O(N)$, where N is the number of signers. However, their multi-signature scheme involves a two-round online signing phase, which increases the computational and communication complexity, making it infeasible for practical IoT applications, especially in scenarios where storage and computational power are limited.

Boschini *et al.* [11] developed a single-round online phase multi-signature scheme based on module-SIS and module-LWE lattice assumptions. This scheme ensures low online round complexity and utilizes key aggregation technique for multi-signature. However, in this scheme, each signer performs the "rejection sampling algorithm (RejSamp)" after generating the signature in the "SignOn" algorithm, which results in inefficiency due to the computational costs. Liu *et al.* [38] proposed an identity-based decentralized multi-signature scheme for IoT applications. Their scheme combines identity-based signature (IBS) with the Schnorr scheme, and it operates within the realm of the elliptic curve discrete logarithm problem (ECDLP). However, a significant limitation of their approach is its vulnerability to quantum threats.

Foteini *et al.* [4] developed a streamlined "Boneh–Lynn–Shacham (BLS) multi-signature system" with optimized key aggregation. In their approach, rather than introducing randomness to the combined signatures, they used a one-time randomization process for the public keys by replacing each public key with a fixed randomized version. However, a drawback of their system is that its security relies on the discrete logarithm problem, making it susceptible to quantum attacks.

Jiang *et al.* [29] used a linear identification (ID) system built via R-modules to develop a compact multi-signature scheme based on key and signature. The compiler described in [29], can transform a linear identity (ID)-based scheme into a compact multi-signature system in which the number of signers has no effect on the aggregated public key or signature size. The main benefit of this approach is that it reduces the complex multi-party signature problem to a weakly secure two-party identification problem. In their scheme, the compiler employs ID-based schemes which use the widely utilized Schnorr ID-based scheme, whereas the ring-based ID system is safe under the ring-LWE and ring-SIS assumptions.

Based on the bimodal lattice signature scheme (BLISS), Liang *et al.* [37] presented a lattice-based multi-signature scheme. Based on the difficulty of the Ring-SIS problem, they show that their scheme [37] is safe and immune to quantum computing attacks because of their fundamental lattice assumptions. After that, a bitcoin transaction is used to illustrate its application to blockchain systems, where all signatures on a single transaction involving many users must be validated on the blockchain, which could result with a lot of data, particularly when it comes to jointly owned cash. Their method also mitigates this problem by reducing the combined signature size, which makes it an effective tool for reducing such hazards. Finally, a comparison is made between the existing state-of-art schemes and the proposed scheme in Table 1.

4 SYSTEM MODEL

In this section, we present a system model that includes a network model for the proposed scheme, as well as a threat model for an adversary who can mount various types of attacks.

4.1 Network Model

In the proposed model (as illustrated in Fig. 1), we consider the network components comprising of IoT devices, a gateway node, and a regional server manager (RSM) for each IoT application. The control room being the Key Generation Center (KGC) is responsible for registering the RSMs in the network. The IoT devices are associated

Table 1. Description, advantages and limitations of existing lattice-based multi-signature schemes

Scheme	Applied Techniques and Advantages	Drawbacks/Limitations
Changhse *et al.* [41]	Blockchain technology is implemented in this scheme. It is based on Ring-SIS assumptions. It resists quantum attacks.	Four-round online signing phase produces a huge computational cost to generate the multi-signature, which makes the scheme less effective for real-world IoT applications.
Fleischhacker *et al.* [22]	It is a non-interactive quantum attacks resistant scheme. The hardness is based on Ring-SIS assumptions.	High computational complexity emerges due to the use of homomorphic vector commitment at the time of multi-signature generation. Even the storage cost is high. There is no blockchain technology or blockchain implementation.
Kansal *et al.* [31]	Its storage cost is low and the hardness of this scheme is based on Ring-SIS assumptions. It ensures that the scheme is resistant against quantum attacks.	It has a two-phase online signing process. However, it is not implemented in real-life IoT applications and does not have any blockchain implementation.
Boschini *et al.* [11]	It has single-round online signing phase and also resists quantum attack, since this scheme is based on the hardness of Module-LWE and Module-SIS.	The size of the keys is enormous and blockchain technology is not implemented. Additionally, this scheme has high multi-signature verification cost.
Jiang *et al.* [29]	It reduces a multi-party problem to a weakly secure two-party problem.	Secret key size is high in this scheme. Blockchain implementation is absent, and it does not have any real-life practical IoT applications. Moreover, this scheme has high storage cost.
Liang *et al.* [37]	This approach is based on the bimodal lattice signature approach, and utilizes the blockchain-enabled systems for demonstrating a cryptocurrency transaction. Also, the costs of multi-signature generation and multi-signature verification are not high. This scheme is quantum attacks resistant, since it is based on Ring-SIS assumptions.	This scheme has no real-life IoT applications and it does not have any blockchain implementation.
Our Scheme	The proposed scheme has a single round online-phase along with blockchain implementation. It is quantum attacks resistant, since it is based on Ring-LWE and Ring-SIS assumptions. It is applied in practical IoT-based smart healthcare applications.	Multi-signature generation cost needs to be further reduced.

with various IoT applications, including smart healthcare, smart home, and smart transportation systems, among others. Each IoT application forms a disjoint cluster to alleviate the burden on the access point, referred to as the gateway node. The RSM assumes the responsibility of registering all the associated IoT devices, cluster heads, and gateway nodes. An IoT device signs a message containing sensing information relevant to its corresponding IoT application and broadcasts the message-signature pair to other IoT devices and the cluster head within its cluster. Upon receiving the message-signature pair, other IoT devices validate the signature's domain, and then sign the same message and subsequently broadcast it within the cluster. The cluster head receives all message-signature pairs from the IoT devices within the cluster, and validates each received signature, and generates a multi-signature based on these received valid signatures. Next, it generates a collection of messages and their multi-signature, and sends the message-multi-signature collection to the associated gateway node. The gateway node is deployed within the cluster and is accountable for collecting message-multi-signature collection from the

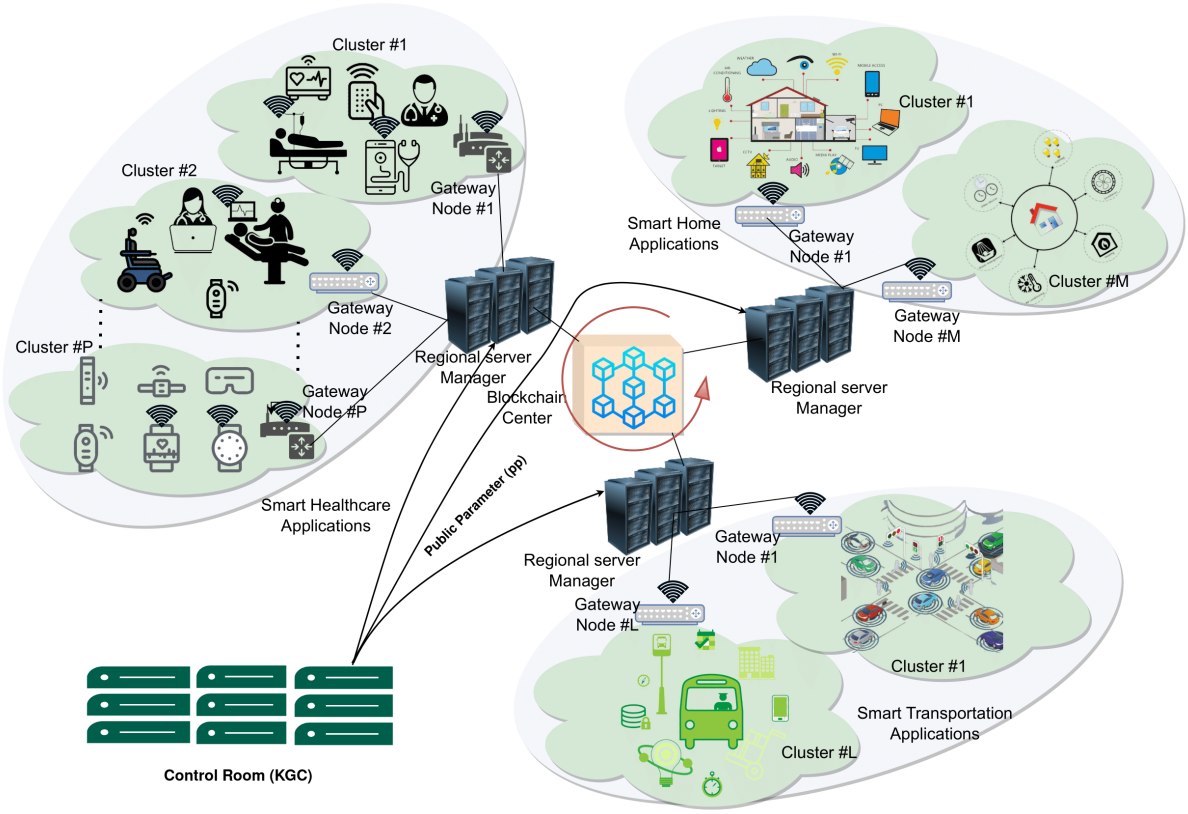

Fig. 1. A network model for IoT applications.

cluster head. It then validates the freshness of the message-multi-signature collections, generates transactions, and forwards such transactions to the RSM. The RSM verifies the received message-multi-signature, adds this transaction into its transaction pool, constructs a block consisting of these transactions, and then includes it into the blockchain using a consensus algorithm. With the extensive storage and computational power of RSMs, the RSMs form a peer-to-peer (P2P) distributed system within the blockchain center.

4.2 Threat Model

In the proposed scheme, the network components, including IoT devices, gateway nodes, and RSM, engage in communication over insecure wireless media. The IoT devices broadcast messages, along with their signatures, to other IoT devices and cluster heads within the same cluster using the insecure channel. Furthermore, cluster heads forward this messages with multi-signature to RSM through the gateway node over a public channel in a similar manner. However, given that the information exchange occurs over an insecure channel, it becomes crucial to address potential security vulnerabilities.

In the proposed scheme, we have taken into consideration a widely-adopted threat model consisting of the Dolev-Yao (DY) [20], Canetti and Krawczyk (CK) [12], Honest-But-Curious (HBC), and extended CK-adversary (eCK) threat models [55], [18], [62].

- Under the DY threat model, an unauthorized individual referred to as an adversary, say $\mathcal{A}$, is capable of eavesdropping on communication messages and can also manipulate, delete, or insert fraudulent content into the communication channel.
- Under the CK-adversary model, $\mathcal{A}$ gains additional capabilities by hijacking the communicated messages. Consequently, not only $\mathcal{A}$ can delete, modify, or insert fake contents, but it can also expose the short-term and long-term secrets that contribute to constructing the signature by compromising a session state.
- The extended CK-adversary (eCK) threat model is a variation of the traditional CK-adversary model. In the eCK model, the adversary $\mathcal{A}$ may possess additional powers or capabilities, making it a more formidable adversary compared to the traditional CK model. These additional capabilities might include the ability to actively execute possible query sequences (for example, a session key reveal query on a session ID, sid) to maintain the session's freshness. In other words, if a session sid or its corresponding session sid^* in the eCK model is not clean, then the session is considered exposed by $\mathcal{A}$. Therefore, the eCK model equips $\mathcal{A}$ with more significant capabilities for disrupting or compromising communication.
- In the HBC threat model, $\mathcal{A}$ is assumed to follow the protocol instructions honestly, meaning it does not deviate from the specified protocol steps or engage in any malicious behavior, such as attempting to break the encryption or tamper with the data being transmitted. However, $\mathcal{A}$ can be "Curious" implying that $\mathcal{A}$ is interested in learning as much as possible from the information that is legitimately accessible to them within the protocol's rules. In other words, $\mathcal{A}$ is curious about the data being exchanged and will attempt to gather as much information as they can within the boundaries of the protocol.

It is worth noticing that the CK-adversary model is not sufficient in order to validate the resilience against "Key Compromise Impersonation (KCI)" and "Known Session Specific Temporary Information Attack (KSTIA)" attacks [16], [35]. On the other side, the eCK model guarantees the property of maximum-exposure resilience by allowing an adversary to access any non-trivial combination of the static or ephemeral private keys of the involved peers, even during the test session. However, the model restricts the leakage of intermediate computations associated with any session [16].

Due to the hostile environment of IoT applications, it is not possible for 24/7 physical monitoring of the deployed IoT devices. In such scenarios, the adversary $\mathcal{A}$ can physically seize some IoT devices and launch side-channel attacks (consequently, quantum side-channel attacks), such as power analysis attacks [43], to extract information stored in the compromised device's memory, and $\mathcal{A}$ is also allowed to launch a lattice-reduction attack to search for a short vector and recover the secret keys as well [9]. Finally, we assume that the gateway nodes and regional server managers will not be physically compromised, and they may be put under physical locking systems [8], [58].

5 PROPOSED LATTICE-BASED SINGLE ROUND ONLINE COLLABORATIVE MULTI-SIGNATURE SCHEME: LBCMS

In this section, we discuss the proposed "lattice-based single round online collaborative multi-signature scheme (LBCMS)". We utilize the various notations from Table 2 for describing the proposed LBCMS. LBCMS contains six algorithms, namely a) Setup, b) KeyGen, c) KeyAgg, d) One-Time Key Generation, e) MSign (Online) and f) MVerify, which are described below in details.

5.1 Setup

This algorithm requires the following steps:

- This algorithm takes an input as the security parameter λ. The trusted authority KGC selects a degree of an irreducible polynomial f, and a large prime q ($q \geq 2^{160} + 21505$), where f is a power of 2 and $q \equiv 1 \pmod{2f}$.

Table 2. Notations and their meanings

Symbol	Description
KGC	Trusted key generation center
pp	Public parameter generated by KGC
f, σ	Power of 2 together with degree of the irreducible polynomial, a Gaussian parameter ≥ 0
bin_q	A binary decomposition function
R_q	A finite field of the type: $\mathbb{Z}_q[x]/< x^f + 1 >$, where $\mathbb{Z}_q = \{0, 1, \cdots, q-1\}$, f is the highest degree of the polynomial, and $q \equiv 1 \pmod{2f}$
$[N]$	A set containing the elements $\{1, 2, \cdots, N\}$
a, q, n	A polynomial chosen randomly from R_q, Large prime number such that $q \equiv 1 \pmod{2f}$, Total number of Regional Manager Servers
N	Total number of signers
ID_i	Identity of the i^{th} signer, i.e., a positive integer $< \sqrt{q}, \forall i \in [N]$
$R_{q,d}$	Sub-field of R_q such that all the co-efficients of each polynomial of this sub-field are in $[-d, d]$, where $d < \sqrt{q}$
i, ϖ	A positive integer in $[N]$, A positive integer $\geq f$.
$R_{q,\theta_\phi.d-1024}$	A sub-field of R_q such that all the co-efficients of each polynomial of this sub-field are in $[-(\theta_\phi.d - 1024), (\theta_\phi.d - 1024)]$
$R_{q,N(\lceil \log q \rceil.d-1024)}$	A sub-field of R_q such that all the co-efficients of each polynomial of this sub-field are in $[-N.(\lceil \log q \rceil.d - 1024), N.(\lceil \log q \rceil.d - 1024)]$
κ	A positive integer
PK	A set consisting of all public keys $\{T_1, T_2, \cdots, T_N\}$
C_κ	A set of polynomials of degree at most $f-1$ from R that have all 0 co-efficients except at most κ co-efficients that are either +1 or −1 together with sup-norm of such polynomial is 1
$(s_{i,1}, s_{i,2})$	Secret key of the i^{th} signer.
T_i, AGK	Public key of the i^{th} signer, aggregated public key
$\phi, Z_i = (z_{1,i}, z_{2,i})$	A message on which all the IoT devices will sign; A pair of signatures generated by the i^{th} signer on the message ϕ.
$Z_\phi = (z_{1,\phi}, z_{2,\phi})$	Multi-signature corresponding to the message ϕ
$H_1(\cdot)$	A "collision-resistant cryptographic one-way hash function which maps an arbitrary string to an element in C_{32}"
$H_2(\cdot)$	A "collision-resistant cryptographic one-way hash function which maps from an arbitrary string to an element in to $Z_q - \{0\}$"
$H_3(\cdot)$	A "collision-resistant cryptographic one-way hash function which maps from $\{0,1\}^f$ to $R_{q,d}$"
$H_4(\cdot)$	A "collision-resistant cryptographic one-way hash function which maps from R_q to S_1"
$H_5(\cdot)$	A "collision-resistant cryptographic one-way hash function different from $H_1(\cdot)$, which maps an arbitrary string to an element in C_{32}"
r, k	The width and height of a random matrix A
A^t	Transposition of a matrix or vector A

- The KGC then chooses a uniformly at random from R_q, a positive integer N, five collision-resistance hash functions H_1, H_2, H_3, H_4 and H_5, a binary decomposition function bin_q, a set of identities for the signers as $\{ID_1, ID_2, \cdots, ID_N\}$, where $\forall u \in [N]$, and ID_u is a positive integer $\leq \sqrt{q}$, the Gaussian parameter $\sigma \geq 0$, and a positive integer $d \leq \sqrt{q}$ together with $0 < d < \frac{q-1}{2}$.
- Finally, the KGC generates the output as the public parameters $pp = \{a, H_1, H_2, H_3, H_4, H_5, q, d, \sigma, f, (ID_u)_{u=1}^N, bin_q\}$.

5.2 KeyGen

This algorithm is executed by the i^{th} signer, say $Signer_i$, for $i \in [N]$. It has the following steps:

- For each $i \in [N]$, $Signer_i$ generates its corresponding private and public keys pair as (sk_i, pk_i) using the following step.
- For each i, $Signer_i$ chooses a pair of random and unique polynomials as $s_i = (s_i^1, s_i^2) \in_R R_q \times R_q$ and then computes $s_i^j = (2.ID_i + 1).\bar{s}_{i,j} + \hat{s}_{i,j}, \forall j \in \{1, 2\}$. Next, $Signer_i$ computes $H_4(\hat{s}_{i,j}) = s_{i,j}, \forall j \in \{1, 2\}$ and $T_i = (a, 1).(s_{i,1}, s_{i,2})^t$, and publishes its public key as $pk_i = T_i$ and keeps its own private key as $sk_i = (s_{i,1}, s_{i,2})$ as secret.

5.3 KeyAgg

This algorithm is executed by the i^{th} signer, $Signer_i$, for each $i \in [N]$ using the following steps:

- Initially, $Signer_i$ considers the set of public keys as $PK = \{T_1, T_2, \cdots, T_N\}$.
- After that, $Signer_i$ generates the aggregation coefficients Ω_v of other signers as well as its own aggregation coefficient Ω_i as $\Omega_v = H_1(T_v, PK), \forall v \in [N] - \{i\}$ and $\Omega_i = H_1(T_i, PK)$.
- Finally, each $Signer_i$ generates the aggregated key as $AGK = \sum_{u=1}^{N} \Omega_u.T_u \pmod{x^f + 1}$.

5.4 One-Time Key Generation

This algorithm is again executed by $Signer_i$, for each $i \in [N]$ using the following steps:

- $Signer_i$ secretly chooses a pair of polynomials $(y_1^i, y_2^i) \in_R R_q \times R_q$.
- $Signer_i$ computes $bin_q(y_1^i) = (\sum_{j=0}^{f-1} y_{1,j,0}^i.x^j, \cdots, \sum_{j=0}^{f-1} y_{1,j,\lceil \log q \rceil - 1}^i.x^j)$ and $bin_q(y_2^i) = (\sum_{j=0}^{f-1} y_{2,j,0}^i.x^j, \cdots, \sum_{j=0}^{f-1} y_{2,j,\lceil \log q \rceil - 1}^i.x^j)$.
- Next, $Signer_i$ computes $(H_3(\sum_{j=0}^{f-1} y_{1,j,0}^i.x^j), \cdots, H_3(\sum_{j=0}^{f-1} y_{1,j,\lceil \log q \rceil - 1}^i.x^j)) = (\mathcal{L}_{1,0}^i, \cdots, \mathcal{L}_{1,\lceil \log q \rceil - 1}^i) \in R_{q,d}^{\lceil \log q \rceil}$, $(H_3(\sum_{j=0}^{f-1} y_{2,j,0}^i.x^j), \cdots, H_3(\sum_{j=0}^{f-1} y_{2,j,\lceil \log q \rceil - 1}^i.x^j)) = (\mathcal{L}_{2,0}^i, \mathcal{L}_{2,1}^i, \cdots, \mathcal{L}_{2,\lceil \log q \rceil - 1}^i) \in R_{q,d}^{\lceil \log q \rceil}$, and also executes $(w_0^i = a.\mathcal{L}_{1,0}^i + \mathcal{L}_{2,0}^i \pmod q, w_1^i = a.\mathcal{L}_{1,1}^i + \mathcal{L}_{2,1}^i \pmod q, \cdots, w_{\lceil \log q \rceil - 1}^i = a.\mathcal{L}_{1,\lceil \log q \rceil - 1}^i + \mathcal{L}_{2,\lceil \log q \rceil - 1}^i \pmod q) = W_i$.
- $Signer_i$ forwards (W_i, T_i) to the other v^{th} signers, where $v \in [N] - \{i\}$ and keeps the secret $\{(y_1^i, y_2^i), (bin_q(y_1^i), bin_q(y_2^i)), (\mathcal{L}_{1,j}^i)_{j=0}^{\lceil \log q \rceil - 1}, (\mathcal{L}_{2,j}^i)_{j=0}^{\lceil \log q \rceil - 1}\}$ with itself, and receives $\{(W_v, T_v) : v \in [N] - \{i\}\}$ from the other v^{th} signers.

Note that the above execution holds for each i^{th}-signer, where $i \in [N]$. Now, each signer computes $(\sum_{j=1}^{N} w_0^j \pmod q, \sum_{j=1}^{N} w_1^j \pmod q, \cdots, \sum_{j=1}^{N} w_{\lceil \log q \rceil - 1}^j \pmod q) = (K_0, K_1, \cdots, K_{\lceil \log q \rceil - 1})$.

5.5 MSign (Online)

This algorithm for signing a message online is executed by each signer. It involves the following steps:

- If there exists an $i \geq 2$ such that $pk_i = pk_1$, the algorithm needs to be restarted.
- Each signer generates his/her associated signature on the message ϕ. To compute the associated signature corresponding to the i^{th} signer on the message ϕ, $Signer_i$ generates $H_2(\{K_j\}_{j \in \{0,1,\cdots,\lceil \log q \rceil - 1\}}, \phi, AGK) = l^\phi \in Z_q - \{0\}$ and computes $bin_q(l^\phi) = (l_0^\phi, l_1^\phi, \cdots, l_{\lceil \log q \rceil - 1}^\phi) \in \{0, 1\}^{\lceil \log q \rceil}$, where ϕ is the message to be signed and the number of 1's in $bin_q(l^\phi)$ is taken as θ_ϕ.
- After that, the i^{th} signer, $Signer_i$ computes $W_\phi = \sum_{j=0}^{\lceil \log q \rceil - 1} (l_j^\phi.K_j)$ and $\hat{y}_{1,i} = bin_q(l^\phi).(\mathcal{L}_{1,j}^i)_{j=0}^{\lceil \log q \rceil - 1} = \sum_{j=0}^{\lceil \log q \rceil - 1} (l_j^\phi.\mathcal{L}_{1,j}^i)$ together with $\hat{y}_{2,i} = bin_q(l^\phi).(\mathcal{L}_{2,j}^i)_{j=0}^{\lceil \log q \rceil - 1} = \sum_{j=0}^{\lceil \log q \rceil - 1}.(l_j^\phi.\mathcal{L}_{2,j}^i)$ and $c_\phi = H_5(W_\phi, \phi,$

AGK), and generates the corresponding signature $Z_i = (z_{1,i}, z_{2,i}) = (c_\phi.\Omega_i.s_{i,1} + \hat{y}_{1,i}, c_\phi.\Omega_i.s_{i,2} + \hat{y}_{2,i})$ for the message ϕ.

- Now, $Signer_i$ verifies if $Z_i = (z_{1,i}, z_{2,i}) \in R_{q,\theta_\phi.d-1024} \times R_{q,\theta_\phi.d-1024}$. If it is not valid, $Signer_i$ rejects the signature and restarts the algorithm. Note that the probability of $Z_i \in R_{q,\theta_\phi.d-1024} \times R_{q,\theta_\phi.d-1024}$ is $(1 - \frac{2048}{2\theta_\phi.d+1})^{2|N|f}$. Otherwise, $Signer_i$ accepts the signature and broadcasts the signature Z_i on the message ϕ to other signers and also receives the signature on the message ϕ as $\{Z_v = (z_{1,v}, z_{2,v}) | v \in [N] - \{i\}\}$ from the remaining v^{th} signers, where $v \in [N] - \{i\}$. After that, $\forall v \in [N] - \{i\}$, the i^{th} signer verifies whether $Z_v \in R_{q,\theta_\phi.d-1024} \times R_{q,\theta_\phi.d-1024}$ or not. If it is verified successfully for each $v \in [N] - \{i\}$, it is taken as an authenticated signature on the message ϕ.

If all the above steps hold successfully for each signer, the designated signer finally computes the multi-signature on the common message ϕ as $Z_\phi = \sum_{j=1}^{N} Z_j = (z_{1,\phi}, z_{2,\phi})$, and sends the signed multi-signature $\{AGK, W_\phi, \phi, c_\phi, Z_\phi\}$ to the verifier, say Ver.

5.6 MVerify

In this algorithm, the verifier Ver will check the validity of the multi-signature Z_ϕ on the received message ϕ. The following steps are followed in this algorithm:

- The input parameters including all the public parameters include $pp = (a, H_1, H_2, H_3, H_4, H_5, q, d, \sigma, f, (ID_u)_{u=1}^{N}, bin_q)$ and the signed multi-signature $\{AGK, W_\phi, \phi, c_\phi, Z_\phi\}$.
- Ver computes $c'_\phi = H_5(W_\phi, \phi, AGK)$ using the message ϕ and then checks whether $c'_\phi = c_\phi$. If it is not equal, then the signature is treated as invalid.
- Ver further verifies if $Z_\phi = (z_{1,\phi}, z_{2,\phi}) \in R_{q,N.(\lceil \log q \rceil.d-1024)} \times R_{q,N.(\lceil \log q \rceil.d-1024)}$. If it is not verified successfully, the signature is treated as invalid.
- Finally, Ver checks whether $(a, 1).Z_\phi = (a, 1).(z_{1,\phi}, z_{2,\phi})^t = c_\phi.AGK + W_\phi$ holds or not. If it is valid, the multi-signature is considered as valid; otherwise, the signature is treated as invalid.

6 APPLYING PROPOSED LBCMS IN BLOCKCHAIN-ENABLED IOT APPLICATIONS

In this section, we discuss how the proposed basic scheme (LBCMS), described in Section 5 is used in practical Internet of Things (IoT) applications (as shown in Fig. 1). The various components in the network involve IoT devices, gateway nodes, regional manager servers, and a Control Room (acting as key generation center,). The *RMS*s form a peer-to-peer distributed blockchain network, known as the Blockchain Center (BC). The purpose of the BC is to receive transactions from the *GN*s and add them to a block, which will be further mined into the blockchain center using the Practical Byzantine Fault Tolerance (PBFT) consensus algorithm [13]. The main purposes of using blockchain-based storage as compared to non-blockchain storage, for example, semi-trusted cloud services are as follows [46]. The blockchain data is immutable, whereas cloud computing data can be altered. Blockchain ensures data integrity without relying on a central authority or third-party validation, while cloud computing does not guarantee full data security or encryption. Blockchain operates on decentralization, avoiding the storage of data in a single location, whereas cloud computing follows a centralized model for data access. In blockchain, data accessibility is crucial, while in cloud computing, data can be either public or private, allowing it to be shared with or hidden from other users. Because the cloud servers are treated as semi-trusted entities, the possibility of several insider attackers within the cloud servers arise. In the context of "Artificial Intelligence/Machine Learning (AI/ML)", we have attacks like "adversarial input attacks" [51], "data poisoning attacks" [28], "model attacks" [23] and "model stealing attacks" [30]. Mitra *et al.* [44] conducted several experiments for big data analytics on IoT data for data poisoning attacks, like "salt noise insertion attack", "Gaussian noise insertion attack", "Poisson noise insertion attack", and "label flipping attack", and observed the following: "when the data input into the blockchain is free from data poisoning attacks, the machine learning

model shows considerably better performance in terms of accuracy, recall, precision, and F1 score, compared to when there is a risk of data poisoning attacks on data stored in a non-blockchain platform". Therefore, we have certain advantages of using the blockchain-based storage as compared to semi-trusted cloud storage in IoT applications.

We consider several IoT applications, including smart healthcare, smart cities, and smart transportation, among others. In an IoT application, say the μ^{th} application, the RMS_μ associated with this application is responsible for registering each network component in offline mode. The RMS_μ along with KGC executes the following phases: 1) public parameter generation phase, 2) registration phase, 3) data collection phase, 4) block formation and addition phase, and 5) dynamic device deployment phase. These phases are discussed in the following subsections.

6.1 Public Parameter Generation Phase

The KGC is considered as a trusted authority who executes the *Setup* phase described in Section 5.1 in order to generate the public parameter set pp and forwards it to all the RMS_π, for $\pi \in [n]$, in the network.

6.2 Registration Phase

Before the deployment of each IoT device IoT_u, where $u \in \{1, 2, 3, \cdots, N_{dr}\}$ in the p^{th} cluster, together with their associated gateway node GN_p , in the μ^{th} application, RMS_μ enrolls them in offline mode. Here, N_{dr} represents the number of IoT devices to be deployed in the p^{th} cluster of the μ^{th} application.

6.2.1 IoT Enrollment Phase. The RMS_μ runs the *KeyGen* algorithm described in Section 5.2 and produces the output $\{(pk_u, sk_u) = (T_u, (s_{u,1}, s_{u,2})); u \in [N_{dr}]\}$. RMS_μ generates $PK_{p,\mu}$ as the collection of $\{pk_u | \ u \in [N_{dr}]\}$ and runs the *KeyAgg* phase mentioned in Section 5.3 to produce the output as $AGK_{p,\mu}$. RMS_μ picks a unique identity ID_u for each IoT device IoT_u in the p^{th} cluser, where $u \in [N_{dr}]$. Before deployment of each IoT_u in the p^{th} cluster of the μ^{th} application, RMS_μ pre-loads registration credentials (sk_u, pk_u, $AGK_{p,\mu}$, H_2, H_3, H_5, ID_u) into its memory. After the successful deployment of each IoT_u in the p^{th} cluster of the μ^{th} application, the RMS_μ removes the set $\{sk_u | u \in [N_{dr}]\}$ from his own database for security reasons and publishes the public parameters $\{PK_{p,\mu}, AGK_{p,\mu}\}$.

The RMS_μ executes the *One-Time Key Generation* phase (described in Section 5.4) in offline mode and runs this phase for one time before receiving the messages.

- The RMS_μ generates $\{(\mathcal{L}^u_{1,j})_{j=0}^{\lceil \log q \rceil - 1}, (\mathcal{L}^u_{2,j})_{j=0}^{\lceil \log q \rceil - 1}, W_u\}$ for each IoT device IoT_u, where $u \in [N_{dr}]$ in the p^{th} cluster of the μ^{th} application. Next, the RMS_μ computes $(K_0, K_1, \cdots, K_{\lceil \log q \rceil - 1})$ using $\{W_u | u \in [N_{dr}]\}$.
- The RMS_μ pre-loads $\{(\mathcal{L}^u_{1,j})_{j=0}^{\lceil \log q \rceil - 1}, (\mathcal{L}^u_{2,j})_{j=0}^{\lceil \log q \rceil - 1}\}$ as the secret credentials and $\{(K_0, K_1, \cdots, K_{\lceil \log q \rceil - 1})\}$ as the public credentials in IoT_u's memory, for each $u \in [N_{dr}]$.

After that, the RMS_μ also removes all the secret credentials $\{\{(y^u_1, y^u_2), (bin_q(y^u_1), bin_q(y^u_2)), (\mathcal{L}^u_{1,j})_{j=0}^{\lceil \log q \rceil - 1}, (\mathcal{L}^u_{2,j})_{j=0}^{\lceil \log q \rceil - 1}\}_{u \in [N_{dr}]}\}$ from its memory for security reasons.

6.2.2 Gateway Node Enrollment Phase. To register the gateway node GN_p associated with the p^{th} cluster in the μ^{th}-application, the RMS_μ generates a unique identity ID_{GN_p}. Next, RMS_μ sends the registration credentials (ID_{GN_p}) to GN_p via secure channel (for instance, in person).

6.3 Data Collection Phase

This phase relies on the following phases.

6.3.1 MSign (Online) Phase. Let an IoT device, say IoT_u, where $u \in [N_{dr}]$ in the p^{th} cluster of the μ^{th} application gathers the sensing data, $Data$, and executes the following steps:

- IoT_u having a message of the form: $\phi = \{Data, ID_u, DTS_u\}$, where DTS_u represents the data collection timestamp generated by the IoT_u. Now, IoT_u executes the *MSign (Online)* phase described in Section 5.5, and generates the associated signature as $(Z_u, c_\phi) = ((z_{1,u}, z_{2,u}), c_\phi)$ on the message ϕ using its own secret key sk_u.
- IoT_u then creates a message-signature pair $MSG_u = \{\phi, (Z_u, c_\phi), CTS_u\}$, where CTS_u represents the current timestamp, and broadcasts it to the remaining IoT devices in the p^{th} cluster of the μ^{th} application domain. For each $v \in [N_{dr}] - \{u\}$, IoT device, IoT_v, receives the message-signature pair MSG_u at the time $CTS_{u,v}$ and checks the validity of the timestamp CTS_u by the condition: $|CTS_u - CTS_{u,v}| < \Delta T$, where ΔT is the "maximum message transmission delay". If it is verified successfully by IoT_v, where $v \in [N_{dr}] - \{u\}$, IoT_v considers MSG_u as the fresh message-signature pair.
- Next, for each $v \in [N_{dr}] - \{u\}$, IoT_v parses Z_u and checks the validity of Z_u by checking whether $Z_u \in R_{q,\theta_\phi.d-1024} \times R_{q,\theta_\phi.d-1024}$ holds or not. If each IoT_v agrees that Z_u is valid, then they accept the signature Z_u as a valid one, and each IoT_v, where $v \in [N_{dr}] - \{u\}$, executes the *MSign (Online)* phase to produce the valid associated signature (Z_v, c_ϕ), and associated message-signature pair $MSG_v = \{\phi, (Z_v, c_\phi), CTS_v\}$ on the same message ϕ, where CTS_v represents the current timestamp associated with IoT_v.
- After receiving all the message-signature pair $\{MSG_v; v \in [N_{dr}] - \{h\}\}$ from other devices in the p^{th} cluster, the IoT device, say IoT_h, being a cluster head of the p^{th} cluster (which is deployed near the GN_p in μ^{th} application domain), verifies all the remaining message-signature pairs $\{MSG_v; v \in [N_{dr}] - \{h\}\}$, and then computes the multi-signature as $Z_\phi = (z_{1,\phi}, z_{2,\phi})$ on the message ϕ by using all the valid signatures $\{Z_u | u \in [N_{dr}]\}$.
- IoT_h generates a *Commit* message, and forwards it to all other IoT_v, for $v \in [N_{dr}] - \{h\}$, belonging to p^{th} cluster. After receiving the *Commit* message, each IoT_v deletes (ϕ, Z_v, c_ϕ) from their memory for security reasons and also for space (memory) constraint of the devices.
- Finally, IoT_h creates message-multi-signature collection $Msg_h^\mu = \{\phi, W_\phi, (c_\phi, Z_\phi), TS_h, AGK_{p,\mu}\}$, where TS_h is the current timestamp associated with message-multi-signature collection, and forwards it to the gateway node GN_p.

6.3.2 Secure Data Aggregation Phase. This phase is executed by the gateway node GN_p and its RMS_μ using the following steps:

- After receiving the message-multi-signature collection, $Msg_h^\mu = \{\phi, W_\phi, (c_\phi, Z_\phi), TS_h, AGK_{p,\mu}\}$ from the IoT_h at a time TS_{h,GN_p}, GN_p verifies the freshness of the message-multi-signature collection Msg_h^μ by using the condition: $|TS_{h,GN_p} - TS_h| < \Delta T$. If it is verified correctly, GN_p generates the current timestamp CTS_{GN_p} and creates a transaction of the form: $TX_p^\mu = (Msg_h^\mu, CTS_{GN_p}, ID_{GN_p})$ and forwards this transaction to its associated RMS_μ.
- After receiving TX_p^μ at time CTS_{GN_p,RMS_μ}, the RMS_μ verifies its freshness by $|CTS_{GN_p,RMS_\mu} - CTS_{GN_p}| < \Delta T$. If the condition is satisfied, the RMS_μ parses the multi-signature Z_ϕ from the message-multi-signature collection Msg_h^μ in this transaction TX_p^μ and runs the *MVerify* algorithm (see Section 5.6) to check whether Z_ϕ lies in $R_{q,N.(\lceil \log q \rceil.d-1024)} \times R_{q,N.(\lceil \log q \rceil.d-1024)}$ or not. If all the executions of the verification are valid, the RMS_μ adds the transaction TX_p^μ in its transactions pool.

6.4 Block Formation and Addition Phase

Once n_t number of valid transactions are stored in the transactions pool, the in-charge of *RMS*, say RMS_μ, starts generating a block, say *Block* and processes that block for mining to add into the *BC* using the PBFT consensus algorithm. A block contains the n_t transactions $(TX_1^\mu, TX_2^\mu, \cdots, TX_p^\mu, \cdots, TX_{n_t}^\mu)$, the multi-signatures in the transactions, a unique block version ($BVer$), the Merkle tree root (MTR), a hash of the previous block in the

chain (PBH), the block creation timestamp (TS_{Block}), the block owner's pseudo-identity, and the current block hash ($CBHash$). The structure of such a block $Block$ is shown in Fig. 2.

Block Header	
Block Version	$BVer$
Previous Block Hash	PBH
Merkle Tree Root	MTR
Timestamp	TS_{Block}
Owner of Block	RMS_{μ}
Block Payload	
List of n_t Transactions containing Message-Multisignatures Collectio	$\{TX_i^{\mu} \mid i = 1, 2, \cdots, n_t\}$
Current Block Hash	$CBHash$

Fig. 2. Structure of a block.

For block addition in the BC, the following steps are executed by the nodes residing in the RMSs peer-to-peer (P2P) blockchain network.

- Let S be chosen as the leader among the RMS_{π}, where $\pi \in \{1, 2, \cdots, n\}$, in a round robin fashion or using some secure leader selection algorithm [61].
- S sends a voting request along with the proposed block, say $Block$, to all of its peer nodes in the P2P network.
- The received block is then verified by each peer node by checking the Merkle tree root, multi-signatures, and current block hash. If all are verified successfully, the peer nodes send a voting reply message with the block verification status as "valid" to the leader S.
- S receives the valid reply messages and counts the verification status of each received reply message by setting an initial value of the counter Ctr_S as 0, and then setting Ctr_S as $Ctr_S = Ctr_S + 1$ for each of the peer nodes' valid voting reply message.
- Now, S determines whether Ctr_S is greater than a predetermined threshold value, $2 \times f_{RMS} + 1$, where f_{RMS} represents the number of (Byzantine) faulty nodes in the RMS P2P network out of the total number of n nodes. If this is the case, S adds the $Block$ to its own distributed ledger and sends a "commit response" message to all of its (peer) follower nodes. The peer nodes finally add the same block, $Block$, to their respective ledgers after receiving S's "commit response" message.

6.5 Dynamic Device Deployment Phase

The *KeyGen* and *One-Time Key Generation* algorithms need to be executed in the offline phase in order to add a new IoT smart device, say IoT_{new} by its in-charge RMS_{μ} corresponding to the p^{th} cluster in the μ^{th} application.

RMS_{μ} creates a secret key sk_{new} and the corresponding public key $pk_{new} = T_{new}$, and other credentials like W_{new} and $\{(\mathcal{L}_{1,j}^{new})_{j=0}^{\lceil \log q \rceil - 1}, (\mathcal{L}_{2,j}^{new})_{j=0}^{\lceil \log q \rceil - 1}\}$ for IoT_{new} in the p^{th} cluster of the μ^{th} application. Then, RMS_{μ} updates the set $PK_{p,\mu}$ as $PK_{p,\mu} \cup \{T_{new}\} = {PK_{p,\mu}}^{new}$, and computes the aggregate key ${AGK_{p,\mu}}^{new} = AGK_{p,\mu} + T_{new}.\Omega_{new}$. RMS_{μ} also computes $(K_0^{new}, K_1^{new}, \cdots, K_{\lceil \log q \rceil - 1}^{new})$ using $(K_0, K_1, \cdots, K_{\lceil \log q \rceil - 1})$ and W_{new}, and forwards $\{(K_0^{new}, K_1^{new}, \cdots, K_{\lceil \log q \rceil - 1}^{new}), {AGK_{p,\mu}}^{new}\}$ to the existing IoT devices deployed under the p^{th} cluster in the μ^{th} application and uploads $(sk_{new}, pk_{new}, {AGK_{p,\mu}}^{new}, (K_0^{new}, K_1^{new}, \cdots, K_{\lceil \log q \rceil - 1}^{new}), (\mathcal{L}_{1,j}^{new})_{j=0}^{\lceil \log q \rceil - 1}, (\mathcal{L}_{2,j}^{new})_{j=0}^{\lceil \log q \rceil - 1})$ into IoT_{new}'s memory.

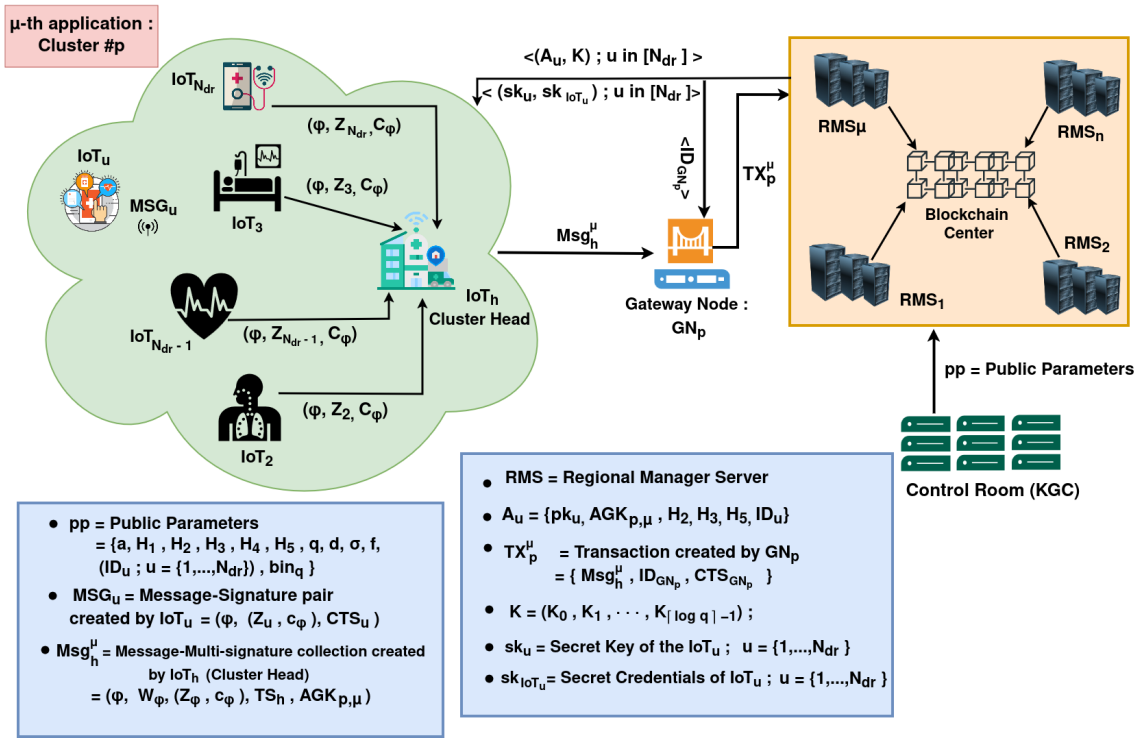

Fig. 3. The proposed LBCMS in action for blockchain-enabled IoT applications.

The existing IoT devices in the μ^{th} application need to update the credentials with the new ones: $\{(K_0^{new}, K_1^{new}, \cdots, K_{\lceil \log q \rceil -1}^{new}), AGK_{p,\mu}{}^{new}\}$. Note that IoT_{new} stores $\{sk_{new}, (\mathcal{L}_{1,j}^{new})_{j=0}^{\lceil \log q \rceil -1}, (\mathcal{L}_{2,j}^{new})_{j=0}^{\lceil \log q \rceil -1}\}$ as secret and $\{PK_{p,\mu}{}^{new}, AGK_{p,\mu}{}^{new}, (K_0^{new}, K_1^{new}, \cdots, K_{\lceil \log q \rceil -1}^{new})\}$ as public.

In Fig. 3, we have shown the proposed LBCMS in action for the blockchain-enabled IoT applications.

7 SECURITY ANALYSIS

This section provides the formal security proof of the proposed signature scheme (LBCMS) under the standard model and also the informal (non-mathematical) security analysis. Moreover, we provide the correctness proof of the signature on the corresponding message by a verifier.

7.1 Correctness Proof

During the *MVerify* phase discussed in Section 5.6, the verifier Ver checks whether $AZ_\phi = (a, 1).(z_{1,\phi}, z_{2,\phi})^t = c_\phi.AGK + W_\phi$ holds or not. If it is valid, the multi-signature is considered as valid; otherwise, the signature is treated as invalid. Now, we have,

$$
\begin{aligned}
&AZ_\phi \\
&= (a, 1).(z_{1,\phi}, z_{2,\phi})^t \\
&= a.z_{1,\phi} + z_{2,\phi} \\
&= \textstyle\sum_{j=1}^{N} (a.z_{1,j} + z_{2,j})
\end{aligned}
$$

$= \sum_{j=1}^{N} c_\phi.\Omega_j.(a.s_{1,j} + s_{2,j}) + \sum_{j=1}^{N}(a.\hat{y}_{1,j} + \hat{y}_{2,j}).$
$= c_\phi(\sum_{j=1}^{N}(\Omega_j.T_j)) + \sum_{j=1}^{N}(\sum_{u=0}^{\lceil \log q \rceil - 1}(a.\mathcal{L}_{1,u}^{j} + \mathcal{L}_{2,u}^{j}).l_u^\phi)$
$= c_\phi.AGK + \sum_{j=1}^{N}(\sum_{u=0}^{\lceil \log q \rceil - 1} w_u^j.l_u^\phi)$
$= c_\phi.AGK + \sum_{u=0}^{\lceil \log q \rceil - 1}\{(\sum_{j=1}^{N} w_u^j).l_u^\phi\}$
$= c_\phi.AGK + \sum_{u=0}^{\lceil \log q \rceil - 1} K_u.l_u^\phi$
$= c_\phi.AGK + W_\phi.$

Hence, the signature is valid.

7.2 Formal Security Analysis

In Theorem 7.1, we provide the formal security of the proposed LBCMS.

Theorem 7.1. *In the random oracle model, suppose there exists a polynomial-time forger, say $\mathcal{F}$, who makes q_H number of hash queries for each hash function H_1, H_2 and H_5, and q_S number of signature queries together with the honest signer involving at most N public keys, and succeeds in providing a forgery of the proposed multi-signature scheme (LBCMS) with a probability δ. Then, there exists an algorithm $\mathcal{G}$ with the same time complexity that can find non-zero vectors $o_1, o_2 \in R_q$ such that $a \cdot o_1 + o_2 = 0$, for a given $A = (a, 1) \leftarrow R_q \times \{1\}$, with $|o_i|_\infty \leq 256 \cdot N \cdot (\lceil \log q \rceil \cdot d - 1024)$ and the probability at least $(\delta - \triangle) \cdot ((\delta - \triangle)/q_T - \frac{1}{|Range_{H_5}|}) \cdot ((\delta - \triangle).(\frac{\delta - \triangle}{q_T} - \frac{1}{|Range_{H_5}|})/q_T - \frac{1}{|Range_{H_1}|})$, where $q_T = q_H + q_S$ and $\triangle = \frac{3 \cdot (q_H + N \cdot q_s)^2}{q^f}$.*

Proof. The proof is provided in Appendix A. □

7.3 Informal Security Analysis

In this section, we emphasize that the proposed multi-signature scheme (LBCMS) is secure against the following potential attacks.

Proposition 7.2. *LBCMS resists replay attack.*

Proof. Assume that an adversary $\mathcal{F}$ intercepts the messages during the *MSign (Online)* phase for μ^{th} application and gathers the messages-signature pair of IoT_u or IoT_h. The message-signature pair for IoT_u is $MSG_u = \{\phi, (Z_u, c_\phi), CTS_u\}$. In this attack, the intention of the $\mathcal{F}$ is to send older message to the recipient. Since the message-signature pair contains the fresh timestamp, therefore once the older message is received, the recipient can reject it by checking its freshness. Similarly, for IoT_h, $\mathcal{F}$ also cannot successfully replay an older message. This shows that the proposed LBCMS resists the replay attack. □

Proposition 7.3. *LBCMS resists Man-in-the Middle (MiTM) attack.*

Proof. In MiTM attack, an adversary $\mathcal{F}$ intercepts and tries to modify the message-signature pair of IoT_u: $MSG_u = \{\phi, (Z_u, c_\phi), CTS_u\}$ or the message-multi-signature collection of IoT_h: $Msg_h^\mu = (\phi, W_\phi, (c_\phi, Z_\phi), TS_h, AGK_{p,\mu})$. MSG_u contain the signature and Msg_h^μ contain the multi-signature on the same information: $\phi = \{Data, ID_u, DTS_u)\}$ which is generated by the private key of the sender. To launch this attack, $\mathcal{F}$ can generate its own current timestamp, but cannot have the knowledge of original private signature keys (for example, secret key sk_u of IoT_u) of the senders. Therefore, $\mathcal{F}$ cannot generate a legitimate signature on behalf of the sender that needs to be included in the message without having the private key of the sender. Hence, the proposed LBCMS is secure against MiTM attack. □

Proposition 7.4. *LBCMS is resilient against device impersonation attack.*

Proof. In this attack, an adversary $\mathcal{F}$ on behalf of a legitimate IoT device, say IoT_u, in the p^{th} cluster wants to send an authentic message to the other IoT devices in the p^{th} cluster for multi-signature. To do so, $\mathcal{F}$ needs to generate a genuine message $MSG_u = \{\phi, (Z_u, c_\phi), CTS_u\}$ on behalf of IoT_u. Since this message contains a legitimate signature, $\mathcal{F}$ can generate a timestamp. However, $\mathcal{F}$ cannot generate the original signature on the message ϕ as it needs the secret key sk_u of IoT_u, which is only available to IoT_u. Thus, $\mathcal{F}$ falls to generate such a message, and similarly for IoT_h. Hence, the proposed scheme resists device impersonation attack. □

Proposition 7.5. *LBCMS is resilient against physical device capture attack.*

Proof. Due to hostile environment, it may not be always possible to monitor all the IoT devices. Thus, a registered IoT device, say IoT_u in the p^{th} cluster of the μ^{th} application, may be physically captured by an adversary $\mathcal{F}$. After capturing IoT_u, $\mathcal{F}$ can launch the classical power analysis attacks [43] to extract the stored credentials from this compromised device's memory. Since every IoT device has unique and distinct secret key and identity along with other secret credentials, compromising an IoT device cannot affect the entire IoT network, because $\mathcal{F}$ has no knowledge of the secret keys of other non-compromised IoT devices. Thus, the secure communication with the compromised IoT device are compromised. However, other non-compromised IoT devices still continue to enjoy secure communications with their neighbor nodes. This property is known as "unconditionally secure against node capture attack" [14, 19]. Therefore, the proposed scheme is secure against physical device capture attack. □

Proposition 7.6. *LBCMS is secure against quantum hybrid attacks.*

Proof. The proposed LBCMS is resistant to various standard quantum attacks. Since it is based on the hardness of Ring-LWE as well as Ring-SIS instances, it can resist Quantum Fourier Transform (QFT) attacks and quantum hybrid attacks. The QFT attack exploits the properties of the Fourier transform to break cryptographic systems. For example, QFT can be used to attack the f^{th} degree truncated polynomial ring used in NTRU cryptosystem [25]. However, lattice-based cryptosystems, such as those based on LWE and Ring-LWE, are not vulnerable to QFT attacks due to their mathematical hardness, and also the hardness of lattice problems. LBCMS is based on the hardness of Ring-LWE and Ring-SIS. As a result, it resists QFT attacks. On the other side, the quantum hybrid attack is based on Howgrave-Graham's classical hybrid attack [26], which combines lattice-based techniques such as basis reduction [36] with guessing techniques like brute-force or meet-in-the-middle attacks [3]. Ring-LWE is specifically designed to resist the attacks by both classical and quantum adversaries. Thus, the proposed scheme can resist quantum hybrid attacks. □

Proposition 7.7. *LBCMS is secure against lattice reduction attack.*

Proof. Lenstra, Lenstra, and Lovász proposed an algorithm, known as the *LLL* algorithm [36], which is a polynomial-time lattice basis reduction algorithm. The objective of the *LLL* algorithm is to transform a given basis $Basis = \{\widehat{b_1}, \widehat{b_2}, \cdots, \widehat{b_f}\}$ for an n-dimensional lattice into a better basis. The vectors in the improved basis should be as short as possible, starting with the shortest vector and then gradually increasing its length until the last vector is reached. Additionally, the vectors in the improved basis should be as orthogonal as possible, ensuring that the dot products of the basis vectors are as close to zero as possible. The *LLL* algorithm generates the associated Gram-Schmidt orthogonal basis $Basis^* = \{b_1^*, b_2^*, \cdots, b_f^*\}$ for the vector space spanned by the basis $Basis$. However, it is important to note that $Basis^*$ is not the basis for the f-dimensional lattice generated by $Basis$. This is because the Gram-Schmidt process involves taking linear combinations with non-integral coefficients. A basis $Basis$ is said to be *LLL*-reduced if it satisfies both the "size condition" and the "Lovasz condition". In small dimensions, the *LLL* algorithm comes close to solving the Shortest Vector Problem (SVP) and Closest Vector Problem (CVP). However, in large dimensions, it is not as effective. Since the lattice used in our proposed scheme has a higher dimension, our scheme (LBCMS) resists the lattice basis reduction attack. □

8 PERFORMANCE COMPARISON

In this section, we analyze the performance of the proposed scheme (LBCMS) and then compare its performance with the existing state of art lattice-based multi-signature schemes, such as the schemes of Ma and Jiang [41], Boschini *et al.* [11], Fleischhacker *et al.* [22], Kansal *et al.* [31], Jiang *et al.* [29], and Liang *et al.* [37].

For comparative study, we assume that $N = 5$, $f = 512$, $d = 6554$, $q = 2^{160} + 25051$, $\lceil \log_2 q \rceil = 160$, $r = 3$, $k = 4$, $m = 2kw + 1$, $\tau \in \{15, 21, 24, 26\}$, $\gamma \in \{41, 44, 46\}$, $\alpha = 10$, $\eta = 3$, $\kappa = 2$, $w = \lceil \log_2 q \rceil$, $\sigma_1 = \sigma_b \cdot \sigma_y \cdot \sqrt{f \cdot (2kw+1) \cdot (r+k)}$, $\sigma_b = \frac{2^{(5/2)}}{\sqrt{\pi}} \cdot 2^{2/fk} \cdot f^{3/2} \cdot \sqrt{kw+1}$, $\sigma_y = \frac{2^9}{\pi \cdot \sqrt{\pi}} \cdot 2^{2/fk} \cdot f^2 . q^{k/r+k} \cdot \sqrt{(kw+1) \cdot (2 + f + \log((r+k) \cdot f))}$, $\Sigma = 20$, and $\beta_\sigma = 20$, where $\sigma_1, \sigma_b, \sigma_y$ are used in [11], $\sigma, t > 0$, $\mu \geq \log^2 f$ and $n_t = 5\sigma . f^2 \sqrt{t\mu} \log^6 f$ are used in [29], and the Gaussian distribution $\sigma = 271$ is used in [37]. In the proposed LBCMS, we consider q as a large prime, f is the power of 2, and f as the degree of the irreducible polynomial. N represents the total number of signers, λ is the security parameter, and d is a positive integer which is less than $\sqrt{q}$.

In Table 3, we have compared the bit-sizes for the secret key, public key and aggregated public key among the proposed LBCMS and other existing schemes [11], [22], [29], [31], [37], [41]. The results in this table shows that the proposed LBCMS requires the same number of bits for secret key, public key and aggregated public key as compared to the schemes [31], [41]. However, the proposed LBCMS performs better than the schemes [11], [22].

Table 3. Comparison of secret key, public key and aggregated public key size (in bits)

Scheme	Secret key size (in bits)	Public key size (in bits)	Aggregated public key size (in bits)
[41]	$2f\lceil \log_2 3 \rceil$	$f\lceil \log_2 q \rceil$	$f\lceil \log_2 q \rceil$
[11]	$(r+k).f\lceil \log_2(2\eta+1) \rceil$	$k.f\lceil \log_2 q \rceil$	$k.f\lceil \log_2 q \rceil$
[22]	$2^\tau.\gamma f(\lceil \log_2 3 \rceil + \lceil \log_2(2\beta_s+1) \rceil)$	$2^{\tau+1} f\lceil \log_2 q \rceil$	N/A
[31]	$2f\lceil \log_2 3 \rceil$	$f\lceil \log_2 q \rceil$	$f\lceil \log_2 q \rceil$
[29]	$2f\lceil \log_2(2\sqrt{f}.\sigma+1) \rceil$	$f\lceil \log_2 q \rceil$	N/A
[37]	$2f\lceil \log 2q \rceil$	$f\lceil \log 2q \rceil + q$	N/A
LBCMS	$2f\lceil \log_2 3 \rceil$	$f\lceil \log_2 q \rceil$	$f\lceil \log_2 q \rceil$

Table 4 shows that the multi-signature size (in bits) as well as storage cost (in bits) among the proposed LBCMS and other schemes. It is observed that the multi-signature size (in bits) of the proposed LBCMS is small with respect to the multi-signature size of the schemes [11], [22]. Moreover, the storage cost (in bits) of the proposed LBCMS is significantly less than the schemes [11], [22]. However, the the storage cost (in bits) of the proposed LBCMS is comparable to the schemes [31], [41].

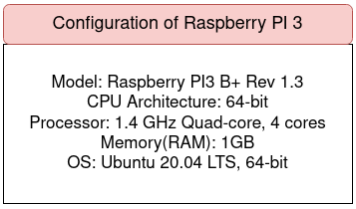

Fig. 4. Raspberry PI3 setting.

Table 4. Comparison of multi-signature size and storage cost (in bits)

Scheme	Multi-signatures size (in bits)	Storage cost (in bits)
[41]	$2f \cdot \lceil \log_2\{2N \cdot (\alpha d - 32) + 1\} \rceil + 160$	20640
[11]	$(r + k) \cdot f \cdot \lceil \log_2(16\sigma_1 + 1) \rceil$	577024
[22]	$f \cdot \gamma \cdot \lceil \log_2(2\beta_\sigma + 1) \rceil$ $+2 \cdot \tau \cdot f \cdot \lceil \log_2 q \rceil \cdot \lceil \log_2(2 \cdot N \cdot \Sigma + 1) \rceil$	19975680
[31]	$2 \cdot f \cdot \lceil \log_2\{2N \cdot (d - 32) + 1\} \rceil$	16384
[29]	$2 \cdot f \cdot \lceil \log_2(2.n_t + 1) \rceil$	103424
[37]	$f \cdot \lceil \log_2(N \cdot \frac{q}{2} + 1) \rceil$	24576
LBCMS	$2 \cdot f \cdot \lceil \log_2\{2N \cdot (\lceil \log_2 q \rceil \cdot d - 32) + 1\} \rceil$	24576

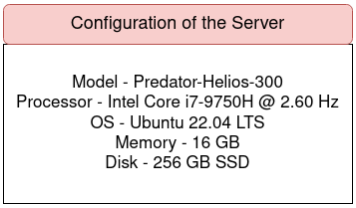

Fig. 5. Server setting.

For computational complexity comparison, we have considered two settings: 1) Raspberry PI 3 setting (as shown in Fig. 4) and 2) server setting (as shown in Fig. 5). In Table 5, we have shown the computational time needed for an IoT device for various cryptographic primitives. Here, the IoT devices are considered under Raspberry PI 3 setting. On the other hand, we have considered the computational time needed for other nodes in the network for various cryptographic primitives in Table 6.

Table 5. Execution time under Raspberry PI 3 setting

Symbols	Operation	Execution time
T_{pmul}	Time required to multiply two $f-1$-degree polynomials ($f = 512$)	63.73 ms
T_{padd}	Time required to add two $f-1$-degree polynomials ($f = 512$)	1.20 ms
T_H	Time required to perform a hash function	0.309 ms
T_{mul}	Time required for performing a scalar multiplication in finite field	0.011 ms
T_{add}	Time required for performing an addition in finite field	0.01 ms
$label$	A function from $\{0,1\}^{\leq \tau}$ to $R_q^{\lceil \log q \rceil}$ defined in [22]	32.68 ms
$T_{vec-mul}$	Time required to dot product of two f-length vectors	10.74 ms

Table 7 indicates that the proposed LBCMS has a lower MSign cost as compared to all other schemes, while Table 8 demonstrates that the MVrfy cost of LBCMS is minimal when compared to the schemes [11], [22]. Moreover, Table 9 provides a comparative study on MSign and MVrfy costs (in seconds) based on the testbed experimental results presented in Tables 5 and 6. It is worth noticing that the proposed LBCMS requires less time as compared to all other schemes, except the scheme [31] for the MSign cost.

Finally, our proposed scheme (LBCMS) relies on the hardness of Ring-LWE and Ring-SIS lattice problems, which demonstrate the resilience against various quantum attacks for LBCMS. Additionally, LBCMS supports a

Table 6. Execution time under server setting

Symbols	Operation	Execution time
T_{pmul}	Time required to multiply two $f-1$-degree polynomials ($f=512$)	15.25 ms
T_{padd}	Time required to add two $f-1$-degree polynomials ($f=512$)	0.069 ms
T_H	Time required to perform a hash function	0.024 ms
T_{mul}	Time required for performing a scalar multiplication in finite field	0.000397 ms
T_{add}	Time required for performing an addition in finite field	0.000321 ms
$T_{vec-mul}$	Time required to dot product of two f-length vectors.	0.37 ms

Table 7. Comparison of multi-signature generation cost (MSign cost)

Scheme	MSign cost
[41]	$(2^{\alpha}.N^2.(\alpha-1)+N.\alpha+2^{\alpha}.N.(N-1)+2^{\alpha+1}.N+2.(N-1)+2^{\alpha+1}.N.(\alpha-1)).T_{padd}+(\alpha+2^{\alpha+1}).N.T_{pmul}+(N.(N-1).\alpha+\alpha.N+N.2^{\alpha}).T_H+(2^{\alpha}.N^2.f.\alpha+2^{\alpha+1}.N.f.\alpha).T_{mul}$
[11]	$\{(r+k).m+k(N+m-2)\}.NT_{padd}+\{(r+k+1)+m.(r+k)+mk\}.NT_{pmul}+\{(r+k)^2+5+(m-1).f\}.NT_{mul}+2NT_H+.\{(m-1).(f-1)+(m-2)\}NT_{add}+(N-1).(r+k).T_{padd}$
[22]	$N.\{(2\gamma+2\tau.\lceil\log_2 q\rceil).(T_{pmul}+T_{padd})+T_H+2\tau.label\}+T_H-(\gamma+2\tau.\lceil\log_2 q\rceil).T_{padd}.$
[31]	$\{4T_{pmul}+(N+2)T_H+(N+2)T_{padd}\}.N+2(N-1)T_{padd}$
[29]	$(\mu+2).T_{pmul}+(5\mu-2).T_{padd}$
[37]	$4.N.T_{pmul}+(4N-1).T_{padd}+N^2.T_H$
LBCMS	$\{2T_H+4.T_{pmul}+3f\lceil\log_2 q\rceil.T_{mul}+(3.\lceil\log_2 q\rceil-1)T_{padd}\}.N+2(N-1)T_{padd}$

Table 8. Comparison of multi-signature verification cost (MVrfy cost)

Scheme	MVrfy cost
[41]	$T_H+2.T_{pmul}+2.T_{padd}$
[11]	$\{(r+k).k.T_{padd}+(r+k+1).k.T_{pmul}+T_H\}+\{(r+k).(f.T_{mul}+(f-1).T_{add})+(r+k-1).T_{add}\}$
[22]	$(\gamma+1+(N+2(\tau-1)).\lceil\log_2 q\rceil).T_{pmul}+((\tau-1).(2\lceil\log_2 q\rceil-1)+\gamma+(N-1).\lceil\log_2 q\rceil).T_{padd}+T_H$
[31]	$T_H+2T_{pmul}+2T_{padd}$
[29]	$2.T_{pmul}+(\mu+1).T_{padd}$
[37]	$4.N.T_{pmul}+(2N+1).T_{padd}+T_H$
LBCMS	$T_H+2T_{pmul}+2T_{padd}$

decentralized architecture and is compatible with blockchain technology. Ultimately, it offers significantly better efficiency and heightened security, making it more resilient to quantum attacks as compared to other compared schemes.

9 BLOCKCHAIN SIMULATION AND RESULTS

In this section, we present the blockchain simulation results by considering the number of servers in the decentralized Peer-to-Peer (P2P) RMS distributed network as 7, where each peer node is considered as a server with configuration: Ubuntu 20.04.6 LTS, 12^{th} Gen Intel® Core™ i7-12800H × 20, Memory 15.3 GiB, Mesa Intel®

Table 9. Comparison of MSign and MVrfy costs (in seconds)

Scheme	MSign cost (in seconds)	MVrfy cost (in seconds)
[41]	3099.96	0.03
[11]	4646.11	0.49
[22]	1584	81.515
[31]	1.34	0.03
[29]	0.6368	0.031
[37]	1.305125	0.306
LBCMS	17.67	0.03

Graphics (ADL GT2)/Mesa Intel® Graphics (ADL GT2), OS type 64-bit, GNOME Version 3.36.8, disk capacity 1.0 TB.

The source code was written using Node.js in Visual Studio Code 2019 [32]. Since blockchain is a distributed technology, adding a block to the chain necessitates a distributed consensus mechanism. Therefore, we employed a voting-based PBFT consensus algorithm for the block mining purpose. The blockchain simulation was conducted under the following two scenarios:

- *Case 1.* In this context, each block contains a fixed number of transactions, specifically 23. The generated blockchain consists of a variable number of blocks, resulting in varying blockchain size. The simulation results depicted in Fig. 6 illustrate the computation time in seconds. This demonstrates the time required to generate a blockchain with different block counts, all of which have the same fixed number of transactions in each block. It is important to note that we employed synthetic data (transactions) within the blocks to assess blockchain time. The findings indicate that as the number of blocks dedicated to blockchain mining increases, the computation time experiences a gradual growth.

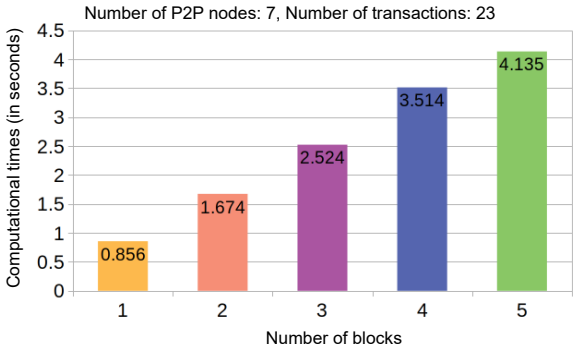

Fig. 6. Blockchain simulation results under Case 1.

- *Case 2.* In this particular case study, we focus on a blockchain structure consisting of a constant number of blocks, precisely 31. However, a distinctive feature lies in the fact that each block can accommodate a different number of transactions. The simulation outcomes presented in Fig. 7 show a clear evidence of the computational time required for the construction of an entire blockchain. It is also observed that there is a gradual increase in time when the number of transactions allocated to each block is increased gradually.

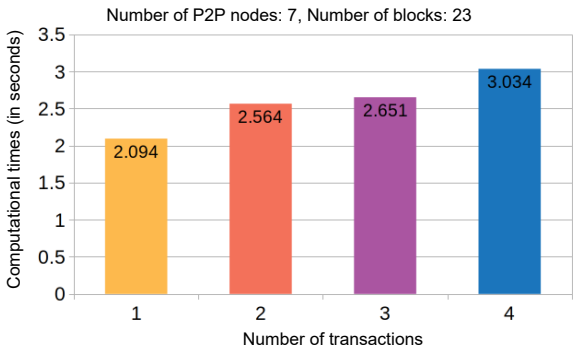

Fig. 7. Blockchain simulation results under Case 2.

10 CONCLUSION AND FUTURE WORKS

In this article, we designed an efficient multi-signature scheme (LBCMS) based on lattice-based cryptography, which leverages on the security of Ring-LWE in conjunction with Ring-SIS lattice assumptions. Next, we applied the LBCMS in real-time IoT applications. In this context, an IoT device within a specific cluster of an application acts as the initial signer, and transmits its message-signature pair to other IoT devices participating in the same cluster of this application. The remaining IoT devices in the cluster of this application play the roles of the independent signers, and generate their individual signatures for the identical message. Subsequently, they relay their individual signatures to the nearest IoT device connected to the gateway node within the specific cluster of this application domain. The nearest IoT devices to the gateway nodes within the applications assume the responsibility of generating the multi-signatures. Subsequently, the collective message, along with the multi-signature, undergoes verification by the RMSs, where the RMSs form blocks with these transactions, and add them into the blockchain. The detailed comprehensive security analysis (both formal and informal) demonstrates the robustness of the proposed scheme, effectively safeguarding against a range of potential threats including quantum attacks. Additionally, we conducted a simulation study within a blockchain framework to assess its impact on computational time. Lastly, when comparing the proposed LBCMS with other existing relevant lattice-based multi-signature schemes, it becomes evident that LBCMS not only provides superior security, but also excels in terms of efficiency.

In the future, our aim is to refine the proposed scheme by making it more streamlined and efficient. This optimization will serve to decrease the computational overhead associated with both generating and verifying lattice-based multi-signatures. This adjustment can also significantly reduce the computational burden involved in verifying and adding blocks to the blockchain through the consensus process.

ACKNOWLEDGEMENTS

The authors would like to thank the anonymous reviewers and associate editor for their valuable feedback on the paper.

REFERENCES

[1] Abderrazak Abdaoui, Aiman Erbad, Abdulla Khalid Al-Ali, Amr Mohamed, and Mohsen Guizani. 2022. Fuzzy Elliptic Curve Cryptography for Authentication in Internet of Things. *IEEE Internet of Things Journal* 9 (2022), 9987–9998.

[2] Prithwi Bagchi, Raj Maheshwari, Basudeb Bera, Ashok Kumar Das, Youngho Park, Pascal Lorenz, and David K. Y. Yau. 2023. Public Blockchain-Envisioned Security Scheme Using Post Quantum Lattice-Based Aggregate Signature for Internet of Drones Applications.

IEEE Transactions on Vehicular Technology (2023), 1–16.

[3] Shi Bai and Steven D. Galbraith. 2014. Lattice Decoding Attacks on Binary LWE. In *Information Security and Privacy*. Wollongong, NSW, Australia, 322–337.

[4] Foteini Baldimtsi, Konstantinos Kryptos Chalkias, Francois Garillot, Jonas Lindstrom, Ben Riva, Arnab Roy, Alberto Sonnino, Pun Waiwitlikhit, and Joy Wang. 2023. Subset-optimized BLS Multi-signature with Key Aggregation. https://eprint.iacr.org/2023/498.

[5] Mihir Bellare and Wei Dai. 2021. Chain Reductions for Multi-signatures and the HBMS Scheme. In *Advances in Cryptology – ASIACRYPT 2021*. Singapore, 650–678.

[6] Mihir Bellare and Gregory Neven. 2006. Multi-Signatures in the Plain Public-Key Model and a General Forking Lemma. In *Proceedings of the 13th ACM Conference on Computer and Communications Security*. New York, NY, United States, 390–399.

[7] Pauline Bert, Pierre-Alain Fouque, Adeline Roux-Langlois, and Mohamed Sabt. 2018. Practical Implementation of Ring-SIS/LWE Based Signature and IBE. In *Post-Quantum Cryptography*. Fort Lauderdale, USA, 271–291.

[8] Elisa Bertino, Ning Shang, and Samuel S. Wagstaff Jr. 2008. An Efficient Time-Bound Hierarchical Key Management Scheme for Secure Broadcasting. *IEEE Transactions on Dependable and Secure Computing* 5, 2 (2008), 65–70.

[9] Sauvik Bhattacharya, Óscar García-Morchón, Ronald Rietman, and Ludo Tolhuizen. 2017. spKEX: An optimized lattice-based key exchange. *IACR Cryptol. ePrint Arch.* 2017 (2017), 709. https://api.semanticscholar.org/CorpusID:35407350.

[10] Joppe W. Bos, Craig Costello, Michael Naehrig, and Douglas Stebila. 2015. Post-Quantum Key Exchange for the TLS Protocol from the Ring Learning with Errors Problem. In *2015 IEEE Symposium on Security and Privacy*. San Jose, California, 553–570.

[11] Cecilia Boschini, Akira Takahashi, and Mehdi Tibouchi. 2022. MuSig-L: Lattice-Based Multi-signature with Single-Round Online Phase. In *Advances in Cryptology – CRYPTO 2022*. Santa Barbara, USA, 276–305.

[12] R. Canetti and H. Krawczyk. 2002. Universally Composable Notions of Key Exchange and Secure Channels. In *International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT'02)*. Amsterdam, The Netherlands, 337–351.

[13] M. Castro and B. Liskov. 2002. Practical Byzantine fault tolerance and proactive recovery. *ACM Transactions on Computer Systems* 20, 4 (2002), 398–461.

[14] S. Chatterjee, A. K. Das, and J. K. Sing. 2014. An Enhanced Access Control Scheme in Wireless Sensor Networks. *Ad Hoc & Sensor Wireless Networks* 21, 1-2 (2014), 121–149.

[15] Flavio Cirillo, David Gómez, Luis Diez, Ignacio Elicegui Maestro, Thomas Barrie Juel Gilbert, and Reza Akhavan. 2020. Smart City IoT Services Creation Through Large-Scale Collaboration. *IEEE Internet of Things Journal* 7 (2020), 5267–5275.

[16] Cas J. F. Cremers. 2009. Formally and Practically Relating the CK, CK-HMQV, and eCK Security Models for Authenticated Key Exchange. *Cryptology ePrint Archive* (2009). https://eprint.iacr.org/2009/253 Paper 2009/253.

[17] Ivan Damgard, Valerio Pastro, Nigel Smart, and Sarah Zakarias. 2012. Multiparty Computation from Somewhat Homomorphic Encryption. In *Advances in Cryptology – CRYPTO 2012*. Santa Barbara, CA, USA, 643–662.

[18] Renu Mary Daniel, Anitha Thomas, Elijah Blessing Rajsingh, and Salaja Silas. 2023. A strengthened eCK secure identity based authenticated key agreement protocol based on the standard CDH assumption. *Information and Computation* 294 (2023), 105067.

[19] A. K. Das. 2012. A random key establishment scheme for multi-phase deployment in large-scale distributed sensor networks. *International Journal of Information Security* 11, 3 (2012), 189–211.

[20] D. Dolev and A. Yao. 1983. On the security of public key protocols. *IEEE Transactions on Information Theory* 29, 2 (1983), 198–208.

[21] Rachid El Bansarkhani and Jan Sturm. 2016. An Efficient Lattice-Based Multisignature Scheme with Applications to Bitcoins. In *Cryptology and Network Security*. Milan, Italy, 140–155.

[22] Nils Fleischhacker, Mark Simkin, and Zhenfei Zhang. 2022. Squirrel: Efficient Synchronized Multi-Signatures from Lattices. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*. Los Angeles, USA, 1109–1123.

[23] X. Gong, Q. Wang, Y. Chen, W. Yang, and X. Jiang. 2020. Model Extraction Attacks and Defenses on Cloud-Based Machine Learning Models. *IEEE Communications Magazine* 58, 12 (2020), 83–89.

[24] Shahriar Hadayeghparast, Siavash Bayat-Sarmadi, and Shahriar Ebrahimi. 2022. High-Speed Post-Quantum Cryptoprocessor Based on RISC-V Architecture for IoT. *IEEE Internet of Things Journal* 9 (2022), 15839–15846.

[25] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. 1998. NTRU: A ring-based public key cryptosystem. In *Algorithmic Number Theory*. Springer Berlin Heidelberg, Berlin, Heidelberg, 267–288.

[26] Nick Howgrave-Graham. 2007. A Hybrid Lattice-Reduction and Meet-in-the-Middle Attack Against NTRU. In *Advances in Cryptology - CRYPTO 2007*. Santa Barbara, California, USA, 150–169.

[27] Ling Hu and Qiang Ni. 2018. IoT-Driven Automated Object Detection Algorithm for Urban Surveillance Systems in Smart Cities. *IEEE Internet of Things Journal* 5 (2018), 747–754.

[28] Hai Huang, Jiaming Mu, Neil Zhenqiang Gong, Qi Li, Bin Liu, and Mingwei Xu. 2021. Data Poisoning Attacks to Deep Learning Based Recommender Systems. *CoRR* abs/2101.02644 (2021). https://arxiv.org/abs/2101.02644

[29] Shaoquan Jiang, Dima Alhadidi, and Hamid Fazli Khojir. 2024. Key-and-Signature Compact Multi-Signatures for Blockchain: A Compiler with Realizations. *IEEE Transactions on Dependable and Secure Computing* (2024), 1–18.

[30] M. Juuti, S. Szyller, S. Marchal, and N. Asokan. 2019. PRADA: Protecting Against DNN Model Stealing Attacks. In *IEEE European Symposium on Security and Privacy (Euro S&P)*. Stockholm, Sweden, 512–527.
[31] Meenakshi Kansal, Amit Kumar Singh, and Ratna Dutta. 2021. Efficient Multi-Signature Scheme Using Lattice. *Comput. J.* 65 (2021), 2421–2429.
[32] Kashish Khullar. 2019. Implementing PBFT in Blockchain. https://medium.com/coinmonks/implementing-pbft-in-blockchain-12368c6c9548.
[33] Handan Kilinc Alper and Jeffrey Burdges. 2021. Two-Round Trip Schnorr Multi-signatures via Delinearized Witnesses. In *Advances in Cryptology – CRYPTO 2021*. 157–188.
[34] An Ngoc Lam, Oystein Haugen, and Jerker Delsing. 2022. Dynamical Orchestration and Configuration Services in Industrial IoT Systems: An Autonomic Approach. *IEEE Open Journal of the Industrial Electronics Society* 3 (2022), 128–145.
[35] Brian LaMacchia, Kristin Lauter, and Anton Mityagin. 2007. Stronger Security of Authenticated Key Exchange. In *Provable Security*, Willy Susilo, Joseph K. Liu, and Yi Mu (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 1–16.
[36] A. K. Lenstra, H. W. Lenstra, and L. Lovász. 1982. Factoring polynomials with rational coefficients. *Math. Ann.* 261, 4 (1982), 515–534.
[37] Xiao Liang, Xiaohui Wang, Qianyi Zhang, Shuai Yuan, and Zhitao Guan. 2023. A Lattice-Based Multisignature Scheme for Blockchain-Enabled Systems. In *Emerging Networking Architecture and Technologies*. Springer Nature Singapore, Shenzhen, China, 336–346.
[38] Han Liu, Dezhi Han, Mingming Cui, Kuan-Ching Li, Alireza Souri, and Mohammad Shojafar. 2023. IdenMultiSig: Identity-Based Decentralized Multi-Signature in Internet of Things. *IEEE Transactions on Computational Social Systems* 10, 4 (2023), 1711–1721.
[39] Ling Lyu, Cailian Chen, Shanying Zhu, Nan Cheng, Bo Yang, and Xinping Guan. 2018. Control Performance Aware Cooperative Transmission in Multiloop Wireless Control Systems for Industrial IoT Applications. *IEEE Internet of Things Journal* 5 (2018), 3954–3966.
[40] Vadim Lyubashevsky. 2012. Lattice Signatures without Trapdoors. In *Advances in Cryptology – EUROCRYPT 2012*. Cambridge, UK, 738–755.
[41] Changshe Ma and Mei Jiang. 2019. Practical lattice-based multisignature schemes for blockchains. *IEEE Access* 7 (2019), 179765–179778.
[42] Gregory Maxwell, Andrew Poelstra, Yannick Seurin, and Pieter Wuille. 2019. Simple schnorr multi-signatures with applications to bitcoin. *Designs, Codes and Cryptography* 87 (2019), 2139–2164.
[43] T. S. Messerges, E. A. Dabbish, and R. H. Sloan. 2002. Examining smart-card security under the threat of power analysis attacks. *IEEE Trans. Comput.* 51, 5 (2002), 541–552.
[44] Ankush Mitra, Basudeb Bera, Ashok Kumar Das, Sajjad Shaukat Jamal, and Ilsun You. 2023. Impact on blockchain-based AI/ML-enabled big data analytics for Cognitive Internet of Things environment. *Computer Communications* 197 (2023), 173–185.
[45] Olga B. Mora-Sanchez, Emmanuel Lopez-Neri, E. Julieta Cedillo-Elias, Emmanuel Aceves-Martinez, and Victor M. Larios. 2021. Validation of IoT Infrastructure for the Construction of Smart Cities Solutions on Living Lab Platform. *IEEE Transactions on Engineering Management* 68 (2021), 899–908.
[46] K. Muzammil. 2024. Blockchain vs Cloud Computing Difference. https://www.aalpha.net/articles/blockchain-vs-cloud-computing-difference/#:~:text=The%20blockchain%20prevents%20data%20tampering,data%20in%20a%20single%20place Accessed on August 2024.
[47] Jonas Nick, Tim Ruffing, and Yannick Seurin. 2021. MuSig2: Simple Two-Round Schnorr Multi-signatures. In *Advances in Cryptology – CRYPTO 2021*. 189–221.
[48] Jonas Nick, Tim Ruffing, Yannick Seurin, and Pieter Wuille. 2020. MuSig-DN: Schnorr Multi-Signatures with Verifiably Deterministic Nonces. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. 1717–1731.
[49] Chris Peikert and Sina Shiehian. 2016. Multi-key FHE from LWE, Revisited. In *Theory of Cryptography*. Beijing, China, 217–238.
[50] Partha Pratim Ray, Dinesh Dash, Khaled Salah, and Neeraj Kumar. 2021. Blockchain for IoT-Based Healthcare: Background, Consensus, Platforms, and Use Cases. *IEEE Systems Journal* 15 (2021), 85–94.
[51] Kui Ren, Tianhang Zheng, Zhan Qin, and Xue Liu. 2020. Adversarial Attacks and Defenses in Deep Learning. *Engineering* 6, 3 (2020), 346–360.
[52] Amit Kumar Sikder, Giuseppe Petracca, Hidayet Aksu, Trent Jaeger, and A. Selcuk Uluagac. 2021. A Survey on Sensor-Based Threats and Attacks to Smart Devices and Applications. *IEEE Communications Surveys & Tutorials* 23, 2 (2021), 1125–1159.
[53] Tianyi Song, Ruinian Li, Bo Mei, Jiguo Yu, Xiaoshuang Xing, and Xiuzhen Cheng. 2017. A Privacy Preserving Communication Protocol for IoT Applications in Smart Homes. *IEEE Internet of Things Journal* 4 (2017), 1844–1852.
[54] Jangirala Srinivas, Ashok Kumar Das, Xiong Li, Muhammad Khurram Khan, and Minho Jo. 2021. Designing Anonymous Signature-Based Authenticated Key Exchange Scheme for Internet of Things-Enabled Smart Grid Systems. *IEEE Transactions on Industrial Informatics* 17, 7 (2021), 4425–4436.
[55] Anusha Vangala, Ashok Kumar Das, Ankush Mitra, Sajal K. Das, and Youngho Park. 2023. Blockchain-Enabled Authenticated Key Agreement Scheme for Mobile Vehicles-Assisted Precision Agricultural IoT Networks. *IEEE Transactions on Information Forensics and Security* 18 (2023), 904–919. doi: 10.1109/TIFS.2022.3231121.
[56] Yuntao Wang, Zhou Su, Jianbing Ni, Ning Zhang, and Xuemin Shen. 2022. Blockchain-Empowered Space-Air-Ground Integrated Networks: Opportunities, Challenges, and Solutions. *IEEE Communications Surveys & Tutorials* 24, 1 (2022), 160–209.

[57] Zhibo Wang, Defang Liu, Yunan Sun, Xiaoyi Pang, Peng Sun, Feng Lin, John C. S. Lui, and Kui Ren. 2022. A Survey on IoT-Enabled Home Automation Systems: Attacks and Defenses. *IEEE Communications Surveys & Tutorials* 24, 4 (2022), 2292–2328.
[58] Mohammad Wazid, Ashok Kumar Das, Vanga Odelu, Neeraj Kumar, and Willy Susilo. 2020. Secure Remote User Authenticated Key Establishment Protocol for Smart Home Environment. *IEEE Transactions on Dependable and Secure Computing* 17, 2 (2020), 391–406.
[59] Mohammad Wazid, Ashok Kumar Das, Sachin Shetty, Prosanta Gope, and Joel J. P. C. Rodrigues. 2021. Security in 5G-Enabled Internet of Things Communication: Issues, Challenges, and Future Research Roadmap. *IEEE Access* 9 (2021), 4466–4489.
[60] Fan Wu, Chunkai Qiu, Taiyang Wu, and Mehmet Rasit Yuce. 2021. Edge-Based Hybrid System Implementation for Long-Range Safety and Healthcare IoT Applications. *IEEE Internet of Things Journal* 8 (2021), 9970–9980.
[61] H. Zhang, J. Wang, and Y. Ding. 2019. Blockchain-based decentralized and secure keyless signature scheme for smart grid. *Energy* 180 (2019), 955–967.
[62] Clement Chan Zheng Wei, Chuah Chai Wen, and Janaka Alawatugoda. 2022. Review on Leakage Resilient Key Exchange Security Models. *International Journal of Communication Networks and Information Security* 11, 1 (2022).

A FORMAL SECURITY ANALYSIS

In Theorem A.2, we provide the formal security of the proposed LBCMS. Before that, we briefly discuss the general forking lemma introduced by [11] to prove the security of our proposed multi-signature scheme.

A.1 General Forking Lemma

The general forking lemma [11] is a fundamental tool used in the analysis of security in cryptographic constructions, particularly in the context of digital signature schemes. It provides a powerful way to reason about the security of a signature scheme by considering the probability of a "fork" occurring in the execution of an adversary.

Lemma A.1. *Assume $q_H \geq 1$ is a fixed integer and consider a set SET with size > 2. Consider a randomized algorithm $\mathcal{M}$, which takes input as ∂ together with $h_1, h_2, \cdots, h_{q_H}$, produces the output as a pair (I, Out), where $I \in [0, q_H]$. Let another randomized algorithm be IG, which be called as the input generator. We define the accepting probability of the randomized algorithm $\mathcal{M}$ represented by $acc_{\mathcal{M}}$, which is defined as the probability that $I \geq 1$ in the experiment provided in Algorithm 1.*

Algorithm 1 Experiment

1: $\partial \leftarrow_R IG$
2: $h_1, h_2, \cdots, h_{q_H} \leftarrow_R SET$
3: $(I, Out) \leftarrow_R \mathcal{M}(\partial, h_1, h_2, \cdots, h_{q_H})$
4: ***return*** (I, Out)

The general forking algorithm $GF_{\mathcal{M}}$ associated with $\mathcal{M}$ is considered as the randomized algorithm which takes input ∂ proceeds as described in Algorithm 2 and outputs either $(1, Out, Out')$ or $(0, \perp, \perp)$. Let $frk = Pr[\hat{b} = 1; \partial \leftarrow_R IG; (\hat{b}, Out, Out') \leftarrow_R GF_{\mathcal{M}}(\partial)]$ and $acc = Pr[\partial \leftarrow_R IG; h_1, h_2, \cdots, h_{q_H} \leftarrow_R SET; (I, Out) \leftarrow_R \mathcal{M}(\partial, h_1, h_2, \cdots, h_{q_H})]$. Then, $frk \geq acc.(\frac{acc}{q_H} - \frac{1}{|SET|})$, where $|SET|$ denotes the cardninality of the set SET.

A.2 Formal Security Proof

Theorem A.2. *In the random oracle model, suppose there exists a polynomial-time forger, say $\mathcal{F}$, who makes q_H number of hash queries for each hash function H_1, H_2 and H_5, and the q_S number of signature queries together with the honest signer involving at most N public keys, and succeeds in providing a forgery of the proposed multi-signature scheme (LBCMS) with a probability δ. Then, there exists an algorithm $\mathcal{G}$ with the same time complexity that can find a pair of non-zero vectors $o_1, o_2 \in R_q$ such that $a \cdot o_1 + o_2 \pmod q = 0$, for a given $A = (a, 1) \leftarrow R_q \times \{1\}$, with*

Algorithm 2 $GF_{\mathcal{M}}(\partial)$

Pick random coins ϱ for the randomized algorithm $\mathcal{M}$
$h_1, h_2, \cdots, h_{q_H} \leftarrow_R SET$
$(I, Out) \leftarrow_R \mathcal{M}(\partial, h_1, h_2, \cdots, h_{q_H}, \varrho)$ using Algorithm 1
if $(I = 0)$ **then**
  return $(0, \perp, \perp)$
end if
$h'_1, h'_2, \cdots, h'_{q_H} \leftarrow_R SET$
$(I', Out') \leftarrow_R \mathcal{M}(\partial, h_1, h_2, \cdots, h_{I-1}, h'_I \cdots h'_{q_H}, \varrho)$
if $(I = I')$ and $(h_I \neq h'_I)$ **then**
  return $(1, Out, Out')$
else
  return $(0, \perp, \perp)$
end if

$|o_i|_\infty \leq 256 \cdot N \cdot (\lceil \log q \rceil \cdot d - 1024)$ *with probability at least* $(\delta - \triangle).((\delta - \triangle)/q_T - \frac{1}{|Range_{H_5}|}).((\delta - \triangle).(\frac{\delta - \triangle}{q_T} - \frac{1}{|Range_{H_5}|})/q_T - \frac{1}{|Range_{H_1}|})$, *where* $q_T = q_H + q_S$ *and* $\triangle = \frac{3.(q_H + N.q_s)^2}{q^f}$.

Proof. We use the Ring-SIS instances, which include $a \in_R R_q$, $q \equiv 1 \pmod{2f}$ and $f = O(\lambda)$, where λ are the security parameter. Let simulator Sim construct the public parameters $pp = (a, H_1, H_2, H_3, H_4, H_5, q, d, \sigma, f, (ID_i)_{i=1}^N, bin_q)$, where $H_1 : \{0,1\}^* \rightarrow C_{32}$, $H_2 : \{0,1\}^* \rightarrow Z_q - \{0\}$, $H_3 : \{0,1\}^f \rightarrow R_{q,d}$, $H_4 : R_q \rightarrow S_1$, $H_5 : \{0,1\}^* \rightarrow C_{32}$ be five cryptographically secure hash functions, d be a positive integer $\leq \sqrt{q}$ and $0 < d < \frac{q-1}{2}$, the bin function $bin_q : R_q \rightarrow R_q^{\lceil \log q \rceil}$, a set of identity of the signer's $\{ID_1, ID_2, \cdots ID_N\}$, where $\forall i \in [N]$, ID_i is a positive integer $\leq \sqrt{q}$ and the Gaussian parameter $\sigma \geq 0$. Here, the hash functions are considered as the random oracles. Let the ranges of the random oracles be defined as $Range_{H_1} = \{v; v \in \{-1, 0, 1\}^f; ||v||_1 \leq 32\}$, $Range_{H_2} = Z_q - \{0\}$, and $Range_{H_5} = \{v; v \in \{-1, 0, 1\}^f; ||v||_1 \leq 32\}$.

An unforgeable game will be played between the forger $\mathcal{F}$ and the simulator Sim. Assume that the forger $\mathcal{F}$ will win the game. The game is now defined as follows.

On the input pp, the responses corresponding to the random oracles queries on H_1, H_5, which are $\lambda_{1,1}, \lambda_{1,2}, \cdots, \lambda_{1,q_H}$ and $\lambda_{5,1}, \lambda_{5,2}, \cdots, \lambda_{5,q_H}$ together with the random coins ϱ_{Sim} that have been selected by the Sim. Then, Sim chooses $s_{i^*} = (s_{i^*}^1, s_{i^*}^2) \in_R R_q \times R_q$ and determines $s_{i^*}^j = (2.ID_i + 1).\bar{s}_{i^*,j} + \hat{s}_{i^*,j}$, $\forall j \in \{1, 2\}$. After that the Sim executes $H_4(\hat{s}_{i^*,j}) = s_{i^*,j}$ $\forall j \in \{1, 2\}$, and sets the public key as $pk_{i^*} = T_{i^*} = (a, 1).(s_{i^*,1}, s_{i^*,2})^t$, and the secret key as $sk_{i^*} = (s_{i^*,1}, s_{i^*,2})$.

Note that the total number of random oracle queries for each H_1, H_2, and H_5 that have been allowed is q_H, and the number of signature queries allowed for the forger $\mathcal{F}$ is q_S. Now, Sim runs the forger $\mathcal{F}$ based on the inputs as the public key T_{i^*}, public parameters pp and random coins ϱ_{Sim}. The hash and the signature queries of the forger $\mathcal{F}$ are simulated by the simulator Sim as follows.

Hash Function Queries: To simulate the random oracle queries, the simulator Sim initially maintains the lists $List_{H_1}$, $List_{H_2}$ and $List_{H_5}$, which consist the response values corresponding to the random oracle queries of H_1, H_2 and H_5, respectively. To count the number of queries of H_1, H_2 and H_5, the simulator Sim keeps the track of three counters ctr_1, ctr_2 and ctr_5.

- H_1-*Query*: Initially, the simulator Sim assigns $H_1(T_i, PK) = \lambda_{1,ctr_1}$. Now, when the query $H_1(T_i, PK)$ is asked, the simulator Sim first checks whether $T_i = T_{i^*}$ holds or not. If it holds, the Sim returns $\perp$; otherwise, it

computes $H_1(T_i, PK)$ and stores the answer corresponding to the content (T_i, PK) into the list $List_{H_1}$ and then sends the answer to the forger $\mathcal{F}$. After storing $H_1(T_i, PK)$ into the list $List_{H_1}$ for each $i \in [N]$, the Sim generates the aggregated public key AGK. If the $\mathcal{F}$ has already asked any random oracle query like $H_2(\cdot, \cdot, AGK)$ or $H_5(\cdot, \cdot, AGK)$ or any such signing query which contains AGK, then an event $Awful_1$ will happen and the simulator Sim will return $\perp$.

- H_2-*Query*: The simulator Sim has a list $List_{H_2}$. After receiving the query $((W_v)_{v\in[N]}, \phi, AGK)$, the simulator Sim first goes through the list $List_{H_2}$ and checks whether the corresponding response is in $List_{H_2}$ or not. If it is present, the Sim returns the associated value from the list $List_{H_2}$ to the forger $\mathcal{F}$; otherwise, the simulator Sim first increases ctr_2 and then honestly computes $H_2((W_v)_{v\in[N]}, \phi, AGK)$ and sends it to the forger $\mathcal{F}$. After that the simulator Sim stores $H_2((W_v)_{v\in[N]}, \phi, AGK)$ into the list $List_{H_2}$.
- H_5-*Query*: The forger $\mathcal{F}$ makes an H_5 query on (W_ϕ, ϕ, AGK) to the simulator Sim. The Sim has a list $List_{H_5}$. Now, Sim first checks the list $List_{H_5}$ and checks whether the corresponding response of $H_5(W_\phi, \phi, AGK)$ is in $List_{H_5}$ or not. If it is there, the simulator Sim returns the associated value from the list $List_{H_5}$ to the forger $\mathcal{F}$; otherwise, the simulator Sim first increases ctr_5 and then honestly computes $H_5(W_\phi, \phi, AGK)$, and sends it to the forger $\mathcal{F}$ and uploads $H_5(W_\phi, \phi, AGK)$ into the list $List_{H_5}$.

Signature Generation Query: The forger $\mathcal{F}$ asks a sign query to the simulator Sim on the message ϕ, with the public key set of signers PK. The simulator Sim answers as follows. Initially, the simulator Sim verifies whether $T_{i^*} \in PK$ or not. If it is not valid, the Sim aborts, and the simulator Sim outputs $\perp$. Otherwise, the simulator Sim parses PK and computes the aggregated public key $AGK = \sum_{j=1}^{N} \Omega_j.T_j \pmod{x^f + 1}$ after querying $H_1(T_j, PK)$ for $1 \leq j \leq N$. Now, the simulator Sim produces the signature Z_{i^*} honestly over the message ϕ by the following way.

The process of the signature generation algorithm are consisting two phases: 1) One-Time Key Generation (OTKG) and 2) MSign (Online). Here, the OTKG is message independent, which will be run only one time. At the beginning, the simulator Sim runs the OTKG as follows.

(1) The simulator Sim chooses secretly a pair of polynomials $(y_1^{i^*}, y_2^{i^*}) \in_R R_q \times R_q$.
(2) Sim computes $bin_q(y_1^{i^*}) = (\sum_{j=0}^{f-1} y_{1,j,0}^{i^*}.x^j, \sum_{j=0}^{f-1} y_{1,j,1}^{i*}.x^j, \cdots, \sum_{j=0}^{f-1} y_{1,j,\lceil \log q\rceil-1}^{i^*}.x^j)$ and $bin_q(y_2^{i^*}) = (\sum_{j=0}^{f-1} y_{2,j,0}^{i^*}.x^j, \sum_{j=0}^{f-1} y_{2,j,1}^{i^*}.x^j, \cdots, \sum_{j=0}^{f-1} y_{2,j,\lceil \log q\rceil-1}^{i^*}.x^j)$.
(3) Sim computes $(H_3(\sum_{j=1}^{f} y_{1,j,0}^{i^*}.x^{j-1}), H_3(\sum_{j=1}^{f} y_{1,j,1}^{i^*}.x^{j-1}), \cdots, H_3(\sum_{j=1}^{f} y_{1,j,\lceil \log q\rceil-1}^{i^*}.x^{j-1})) = (\mathcal{L}_{1,0}^{i^*}, \mathcal{L}_{1,1}^{i^*}, \cdots, \mathcal{L}_{1,\lceil \log q\rceil-1}^{i^*}) \in R_{q,d}^{\lceil \log q\rceil}$ and $(H_3(\sum_{j=0}^{f-1} y_{2,j,0}^{i^*}.x^j), H_3(\sum_{j=0}^{f-1} y_{2,j,1}^{i^*}.x^j), \cdots, H_3(\sum_{j=0}^{f-1} y_{2,j,\lceil \log q\rceil-1}^{i^*}.x^j)) = (\mathcal{L}_{2,0}^{i^*}, \mathcal{L}_{2,1}^{i^*}, \cdots, \mathcal{L}_{2,\lceil \log q\rceil-1}^{i^*}) \in R_{q,d}^{\lceil \log q\rceil}$.
(4) Sim executes $(w_0^{i^*} = a.\mathcal{L}_{1,0}^{i^*} + \mathcal{L}_{2,0}^{i^*} \pmod q, w_1^{i^*} = a.\mathcal{L}_{1,1}^{i^*} + \mathcal{L}_{2,1}^{i^*} \pmod q, \cdots, w_{\lceil \log q\rceil-1}^{i^*} = a.\mathcal{L}_{1,\lceil \log q\rceil-1}^{i^*} + \mathcal{L}_{2,\lceil \log q\rceil-1}^{i^*} \pmod q) = W_{i^*}$.
(5) Sim sends (W_{i^*}, T_{i^*}) to the forger $\mathcal{F}$ and keeps the secrets $\{(y_1^{i^*}, y_2^{i^*}), (bin_q(y_1^{i^*}), bin_q(y_2^{i^*})), (\mathcal{L}_{1,j}^{i^*})_{j=0}^{\lceil \log q\rceil-1}, (\mathcal{L}_{2,j}^{i^*})_{j=0}^{\lceil \log q\rceil-1}\}$ and received $\{(W_v, T_v) : v \in [N] - \{i^*\}\}$ from the the forger $\mathcal{F}$.
(6) Finally, Sim executes $(\sum_{j=1}^{N} w_0^j \pmod q, \sum_{j=1}^{N} w_1^j \pmod q, \cdots, \sum_{j=1}^{N} w_{\lceil \log q\rceil-1}^j \pmod q) = (K_0, K_1, \cdots, K_{\lceil \log q\rceil-1})$.

After running the OTKG, the simulator Sim run the "MSign (online) phase" as follows:

(1) The simulator Sim checks if $\exists$ any $i \geq 2$ such that $pk_i = pk_1$, then abort.
(2) Sim then generates the associated signature by computing $H_2(\{K_j\}_{j\in\{0,1,\cdots,\lceil \log q\rceil-1\}}, \phi, AGK) = l^\phi \in Z_q - \{0\}$ and $bin_q(l^\phi) = (l_0^\phi, l_1^\phi, \cdots, l_{\lceil \log q\rceil-1}^\phi) \in \{0,1\}^{\lceil \log q\rceil}$, assuming the number of 1s in $bin_q(l^\phi)$ is θ_ϕ.

(3) Next, Sim computes $W_\phi = \sum_{j=0}^{\lceil \log q \rceil - 1}(l_j^\phi.K_j)$ and $\hat{y}_{1,i^*} = bin_q(l^\phi).(\mathcal{L}_{1,j}^{i^*})_{j=0}^{\lceil \log q \rceil - 1} = \sum_{j=0}^{\lceil \log q \rceil - 1}(l_j^\phi.\mathcal{L}_{1,j}^{i^*})$ together with $\hat{y}_{2,i^*} = bin_q(l^\phi).(\mathcal{L}_{2,j}^{i^*})_{j=0}^{\lceil \log q \rceil - 1} = \sum_{j=0}^{\lceil \log q \rceil - 1}.(l_j^\phi.\mathcal{L}_{2,j}^{i^*})$.
(4) Sim generates $c_\phi = H_5(W_\phi, \phi, AGK)$ and $\Omega_{i^*} = H_1(T_{i^*}, PK)$. After that the simulator Sim computes the corresponding signature as $Z_{i^*} = (z_{1,i^*}, z_{2,i^*}) = (c_\phi.\Omega_{i^*}.s_{i^*,1} + \hat{y}_{1,i^*} \pmod{x^f+1}, c_\phi.\Omega_{i^*}.s_{i^*,2} + \hat{y}_{2,i^*} \pmod{x^f+1})$.
(5) The simulator Sim verifies whether $Z_{i^*} = (z_{1,i^*}, z_{2,i^*}) \in R_{q,\theta_\phi.d-1024} \times R_{q,\theta_\phi.d-1024}$. If it is not so, then Sim repeats the online phase from Step 1 to Step 5 of the signing algorithm. Otherwise, Sim accepts the signature and forwards the signature $Z_{i^*} = (z_{1,i^*}, z_{2,i^*})$ to the forger $\mathcal{F}$.

Now, if the forger $\mathcal{F}$ outputs $\perp$, the simulator Sim also outputs $\perp$. Otherwise, $\mathcal{F}$ executes the multi-signature $\sigma_\phi^* = (Z_\phi, c_\phi)$ on the message ϕ under the public key set PK, where $Z_\phi = (z_{1,\phi}, z_{2,\phi})$. The simulator Sim first parses $PK = \{T_1 = T_{i^*}, T_2, \cdots, T_N\}$, and then computes $\Omega_j = H_1(T_j, PK)$, the aggregated public key $AGK = \sum_{j=1}^N T_j.\Omega_j = \sum_{j=1}^N T_j.H_1(T_j, PK)$, and checks if the multi-signature is valid. If the signature is valid, the following hold:

i) $Z_\phi = (z_{1,\phi}, z_{2,\phi}) \in R_{q,N.(\lceil \log q \rceil.d-1024)} \times R_{q,N.(\lceil \log q \rceil.d-1024)}$.
ii) $H_5(W_\phi, \phi, AGK) = c_\phi$.
iii) $W_\phi = AZ_\phi$ - $c_\phi.AGK = (a, 1).(z_{1,\phi}, z_{2,\phi})^t$ - $c_\phi.AGK$.

The simulator Sim returns "fail" when the following conditions hold:

i) σ_ϕ^* is an invalid multi-signature.
ii) $T_{i^*} \notin PK$.

Let t denote an index such that the output of $H_1(T_{i^*}, PK)$ be asked in the t^{th} query and g be another index such that $H_5(W_\phi, \phi, AGK) = c_\phi$ occurred in the g^{th} query. The simulator Sim using the inputs as the ring-SIS instance $(a, 1)$, random coins ϱ_{Sim} and the forger $\mathcal{F}$ as a subroutine, outputs $(\{g\}, \{t, , Z_\phi, c_\phi, PK, W_\phi, AGK, \Omega_1, \Omega_2, \cdots, \Omega_N\})$. If not, then the Sim will halt with output $(0, \perp)$. We now discuss the accepting probability of the simulator Sim based on the discussion in the general forking lemma provided in Lemma A.1. The simulation done by the the Sim is indistinguishable from the real environment, if the following events $Awful_1$ and $AKColl$ do not occur.

- $Awful_1$: Suppose the forger $\mathcal{F}$ already knows the aggregated public key AGK before finishing all the computation $H_1(T_i, PK)$. Since, we are assuming $H_1(T_{i^*}, PK) = \lambda_{1,t}$, so $AGK = \lambda_{1,t}.T_{i^*} + \sum_{T_i \in PK; T_i \neq T_{i^*}} \Omega_i.T_i \pmod{x^f+1}$, where $\lambda_{1,t}$ and Ω_i are selected uniformly at random in C_{32}. Thus, AGK is also uniformly random in the set of q^f vectors. Since there are at most $(q_H + N.q_S)$ defined entries in $List_{H_2}$ because the number of queries to H_2 are at most $(q_H + N.q_S)$, and at most $(q_H + N.q_S)$ set of assignments together with the number of defined entries in $List_{H_5}$ are at most $(q_H + N.q_S)$ because the number of queries to H_5 are at most $(q_H + N.q_S)$, so the event $Awful_1$ occurs with the probability $2(q_H + N.q_S)^2/q^f$.
- $AKColl$: Assume that the forger $\mathcal{F}$ obtains the same aggregated public key $AGK = AGK'$ from two different sets of public keys PK and PK'. Since AGK is uniform in a set of q^f ring elements and it is independent from the other aggregated public keys, so the probability that $AKColl$ happens is at most $(q_H + N.q_s)^2/q^f$.

As a result, we have, $acc(Sim) = Pr[$ the forgery $\mathcal{F}$ is valid] - $Pr[Awful_1]$ - $Pr[AKColl]$. Thus, the accepting probability of the simulator Sim becomes $acc(Sim) \geq \delta$ - $3.(q_H + N.q_s)^2/q^f$.

We now build a polynomial time algorithm $\mathcal{D}$ that takes the inputs as $(pp, \lambda_{1,1}, \lambda_{1,2}, \cdots, \lambda_{1,q_H}, \varrho_{\mathcal{D}})$, where $\lambda_{1,1}, \lambda_{1,2}, \cdots, \lambda_{1,q_H}$ represent the random oracle responses of H_1 and $\varrho_{\mathcal{D}}$ denotes the random coins corresponding to the algorithm $\mathcal{D}$. $\mathcal{D}$ runs the general forking algorithm $\mathcal{GF}_{Sim}$, and produces the output as $(g, Out; g', Out')$, where $g = g'$ and $Out = \{t, Z_\phi, c_\phi, PK, W_\phi, AGK, \Omega_1, \Omega_2, \cdots, \Omega_N\}$ and $Out' = \{t', Z'_{\phi'}, c'_{\phi'}, H', W_{\phi'}, AGK', \Omega'_1, \Omega'_2, \cdots, \Omega'_{N'}\}$. Generally, in the two executions from $\mathcal{GF}_{Sim}$, the forking point is located in the g^{th} query of $H_5(W_\phi, \phi, AGK)$, which implies that before this point the environments simulated by the

simulator Sim are the same during the two performances, that is, $\varrho = \{\partial, \lambda_{5,1}, \lambda_{5,2}, \cdots, \lambda_{5,g-1}, \lambda_{5,g}, \lambda_{5,g+1} \cdots, \lambda_{5,q_H}\}$ and $\varrho' = \{\partial, \lambda_{5,1}, \lambda_{5,2}, \cdots, \lambda_{5,g-1}, \lambda'_{5,g}, \lambda'_{5,g+1} \cdots, \lambda'_{5,q_H}\}$. From the g^{th} query of H_5, the random oracle gives the different outputs. All the arguments in the H_5-query are same in the two executions, that is, $AGK = AGK'$, $W_\phi = W_{\phi'}$ and $\phi = \phi'$. The query $H_1(T_i, PK)$ has been queried before the forking point, so $PK = PK'$, that is, $N = N'$ and the responses $H_1(T_i, PK)$ are same in the both executions, that is, $\Omega_i = \Omega'_i, \forall i \in [N]$. Now, based on the output of $\mathcal{GF}_{Sim}$, we have the following results:

$$AZ_\phi - c_\phi.AGK = A.Z'_{\phi'} - c'_{\phi'}.AGK \tag{1}$$

Simplifying Eq. (1), we have:

$$A.(Z_\phi - Z'_\phi) + (c'_\phi - c_\phi).AGK = 0 \tag{2}$$

The algorithm $\mathcal{D}$ produces an output (t, Ot), where $Ot = (Z_\phi, Z'_\phi, c_\phi, c'_\phi, AGK, PK, \Omega_1, \Omega_2, \cdots, \Omega_N)$. The accepting probability of $\mathcal{D}$ is then $acc(\mathcal{D}) \geq acc(Sim).(\frac{acc(Sim)}{q_T} - \frac{1}{|Range_{H_5}|})$, where $acc(Sim)$ represents the acceptance probability of the simulator Sim.

We construct another polynomial time algorithm $\mathcal{G}$ as follows. It takes input as pp, and runs the general forking algorithm $\mathcal{GF}_{\mathcal{D}}$ to produce the output $(t, Ot; t', \tilde{Ot})$ such that $t = t'$ and $Ot = (Z_\phi, Z'_\phi, c_\phi, c'_\phi, AGK, PK, \Omega_1, \Omega_2, \cdots, \Omega_N)$ and $\tilde{Ot} = (\tilde{Z}_{\phi''}, \tilde{Z}'_{\phi''}, \tilde{c}_{\phi''}, \tilde{c}'_{\phi''}, \tilde{AGK}, \tilde{PK}, \tilde{\Omega}_1, \tilde{\Omega}_2, \cdots, \tilde{\Omega}_N)$. In both the runs of $\mathcal{GF}_{\mathcal{D}}$, the forking point is located in the t^{th} query of $H_1(T_{i^*}, PK)$. This means that prior to this point, the simulated environments of the simulator Sim are same in both runs. Consequently, all the arguments of $H_1(T_{i^*}, PK)$ results are identical. Then, $PK = \tilde{PK}$ and $T_{i^*} = \tilde{T_{i^*}}$. Also, each $H_1(T_j, PK)$ $(T_j \neq T_{i^*})$ represents the same value in both runs as per the simulator Sim's description. It ensures that $\Omega_j = \tilde{\Omega}_j$ for $T_j \neq T_{i^*}$ and $\Omega_j \neq \tilde{\Omega}_j$ for $T_j = T_{i^*}$, which in turn also ensures that $AGK \neq \tilde{AGK}$. Based on the outputs of $\mathcal{GF}_{\mathcal{D}}$, we can write $AGK = \sum_{i=1}^{N} \Omega_i.T_i \pmod{x^f + 1}$ and $\tilde{AGK} = \sum_{i=1}^{N} \tilde{\Omega}_i.\tilde{T}_i \pmod{x^f + 1}$. Based on Eq. (2), we can write that

$$\begin{aligned}
A.(Z_\phi - Z'_\phi) + (c'_\phi - c_\phi).AGK &= 0 \\
A.(Z_\phi - Z'_\phi) + (c'_\phi - c_\phi).(\sum_{i=1}^{N} \Omega_i.T_i \pmod{x^f + 1}) &= 0 \\
A.(Z_\phi - Z'_\phi) + A(c'_\phi - c_\phi).(\sum_{i=1}^{N} \Omega_i.sk_i \pmod{x^f + 1}) &= 0 \\
A.[(Z_\phi - Z'_\phi) + (c'_\phi - c_\phi).(\sum_{i=1}^{N} \Omega_i.sk_i \pmod{x^f + 1})] &= 0
\end{aligned} \tag{3}$$

Similarly, we have:

$$\begin{aligned}
A.(\tilde{Z}_{\phi''} - \tilde{Z}'_{\phi''}) + (\tilde{c}'_{\phi''} - \tilde{c}_{\phi''}).\tilde{AGK} &= 0 \\
A.(\tilde{Z}_{\phi''} - \tilde{Z}'_{\phi''}) + (\tilde{c}'_{\phi''} - \tilde{c}_{\phi''}).(\sum_{i=1}^{N} \tilde{\Omega}_i.\tilde{T}_i \pmod{x^f + 1}) &= 0 \\
A.(\tilde{Z}_{\phi''} - \tilde{Z}'_{\phi''}) + A.(\tilde{c}'_{\phi''} - \tilde{c}_{\phi''}).(\sum_{i=1}^{N} \tilde{\Omega}_i.\tilde{sk}_i \pmod{x^f + 1}) &= 0 \\
A.[(\tilde{Z}_{\phi''} - \tilde{Z}'_{\phi''}) + (\tilde{c}'_{\phi''} - \tilde{c}_{\phi''}).(\sum_{i=1}^{N} \tilde{\Omega}_i.\tilde{sk}_i \pmod{x^f + 1})] &= 0
\end{aligned} \tag{4}$$

Simplifying further Eqs. (3) and (4), we obtain:

$$A.[(Z_\phi - Z'_\phi).(\tilde{c}'_{\phi''} - \tilde{c}_{\phi''}) - (\tilde{Z}_{\phi''} - \tilde{Z}'_{\phi''}).(c'_\phi - c_\phi) + (c'_\phi - c_\phi).(\tilde{c}'_{\phi''} - \tilde{c}_{\phi''}).(H_1(T_{i^*}, PK) - H_1(\tilde{T_{i^*}}, \tilde{PK})).sk_{i^*}] = 0 \quad (5)$$

$$A.[(Z_\phi - Z'_\phi).(\tilde{c}'_{\phi''} - \tilde{c}_{\phi''}) - (\tilde{Z}_{\phi''} - \tilde{Z}'_{\phi''}).(c'_\phi - c_\phi) + (c'_\phi - c_\phi).(\tilde{c}'_{\phi''} - \tilde{c}_{\phi''}).(\Omega_1 - \tilde{\Omega}_1).sk_{i^*}] = 0 \quad (6)$$

Since $\|Z_\phi\|_\infty \leq N.(\lceil \log q \rceil.d - 1024)$, $\|Z'_\phi\|_\infty \leq N.(\lceil \log q \rceil.d - 1024)$, $\|\tilde{Z}_{\phi''}\|_\infty \leq N.(\lceil \log q \rceil.d - 1024)$ and $\|\tilde{Z}'_{\phi''}\|_\infty \leq N.(\lceil \log q \rceil.d - 1024)$, and $\|Z_\phi - Z'_\phi\|_\infty \leq 2N.(\lceil \log q \rceil.d - 1024)$ with $\|\tilde{Z}_{\phi''} - \tilde{Z}'_{\phi''}\|_\infty \leq 2N.(\lceil \log q \rceil.d - 1024)$, we obtain $\|[(Z_\phi - Z'_\phi).(\tilde{c}'_{\phi''} - \tilde{c}_{\phi''}) - (\tilde{Z}_{\phi''} - \tilde{Z}'_{\phi''}).(c'_\phi - c_\phi) + (c'_\phi - c_\phi).(\tilde{c}'_{\phi''} - \tilde{c}_{\phi''}).(\Omega_1 - \tilde{\Omega}_1).sk_{i^*}]\|_\infty \leq 256.N.(\lceil \log q \rceil.d - 1024) + 64^3$.

Next, we want to show that

$$(Z_\phi - Z'_\phi).(\tilde{c}'_{\phi''} - \tilde{c}_{\phi''}) - (\tilde{Z}_{\phi''} - \tilde{Z}'_{\phi''}).(c'_\phi - c_\phi) + (c'_\phi - c_\phi).(\tilde{c}'_{\phi''} - \tilde{c}_{\phi''}).(\Omega_1 - \tilde{\Omega}_1).sk_{i^*} \neq 0 \quad (7)$$

Based on the results in [40], we use $sk_\bullet$ that has slightly higher coefficients, so there exists another $sk_{\bullet\bullet}$ such that $A.sk_\bullet = A.sk_{\bullet\bullet}$. Now, if

$$(Z_\phi - Z'_\phi).(\tilde{c}'_{\phi''} - \tilde{c}_{\phi''}) - (\tilde{Z}_{\phi''} - \tilde{Z}'_{\phi''}).(c'_\phi - c_\phi) + (c'_\phi - c_\phi).(\tilde{c}'_{\phi''} - \tilde{c}_{\phi''}).(\Omega_1 - \tilde{\Omega}_1).sk_{i^*} = 0, \quad (8)$$

there exist another $sk_{\bullet\bullet}$ for which

$$(Z_\phi - Z'_\phi).(\tilde{c}'_{\phi''} - \tilde{c}_{\phi''}) - (\tilde{Z}_{\phi''} - \tilde{Z}'_{\phi''}).(c'_\phi - c_\phi) + (c'_\phi - c_\phi).(\tilde{c}'_{\phi''} - \tilde{c}_{\phi''}).(\Omega_1 - \tilde{\Omega}_1).sk_{\bullet\bullet} \neq 0 \quad (9)$$

For the signature generation in the time of simulation, we will get a non-zero answer with probability at least $\frac{1}{2}$, since each key has an equal probability of being chosen. Therefore, it can solve Ring-SIS$_{2f,q,\mathcal{D}}$ if $\mathcal{D} \leq 256.N.(\lceil \log_2 q \rceil.d - 1024) + 64^3$. The accepting probability of the algorithm $\mathcal{G}$ is defined as

$$acc(\mathcal{G}) \geq acc(\mathcal{D}).(\frac{acc(\mathcal{D})}{q_T} - \frac{1}{|Range_{H_1}|}) \geq acc(Sim).(\frac{acc(Sim)}{q_T} - \frac{1}{|Range_{H_5}|})(\frac{acc(\mathcal{D})}{q_T} - \frac{1}{|Range_{H_1}|}) \quad (10)$$

where

$$(\frac{acc(\mathcal{D})}{q_T} - \frac{1}{|Range_{H_1}|}) \geq (\frac{acc(Sim).(\frac{acc(Sim)}{q_T} - \frac{1}{|Range_{H_5}|})}{q_T} - \frac{1}{|Range_{H_1}|}) \quad (11)$$

Using Eqs. (10) and (11), we have:

$$\begin{aligned} acc(\mathcal{G}) &\geq acc(Sim).(\frac{acc(Sim)}{q_T} - \frac{1}{|Range_{H_5}|}).(\frac{acc(Sim).(\frac{acc(Sim)}{q_T} - \frac{1}{|Range_{H_5}|})}{q_T} - \frac{1}{|Range_{H_1}|}) \\ &\geq (\delta - \triangle).((\delta - \triangle)/q_T - \frac{1}{|Range_{H_5}|}).((\delta - \triangle).(\frac{\delta - \triangle}{q_T} - \frac{1}{|Range_{H_5}|})/q_T - \frac{1}{|Range_{H_1}|}), \end{aligned}$$

where $acc(Sim) \geq (\delta - \triangle)$. Hence, the theorem is proved. □