Hindawi
Security and Communication Networks
Volume 2022, Article ID 5717185, 11 pages
https://doi.org/10.1155/2022/5717185

Research Article

A Lamus-Based Flight Data Sharing Model on Consortium Blockchain

Fengyin Li,[1] **Yang Cui**,[1] **Baogui Huang**,[1] **Siqi Yu**,[1] **Peiyu Liu**,[2] **Yilei Wang**,[1] **and Tao Li**[1,3]

[1]*School of Computer Science, Qufu Normal University, Rizhao 276800, China*
[2]*School of Information Science and Engineering, Shandong Normal University, Jinan 250014, China*
[3]*State Key Laboratory of Public Big Data, College of Computer Science and Technology, Guizhou University, Guiyang 550025, China*

Correspondence should be addressed to Tao Li; litao_2019@qfnu.edu.cn

Received 24 February 2022; Revised 29 March 2022; Accepted 18 April 2022; Published 9 May 2022

Academic Editor: Zhijun Wu

Currently, traditional flight data sharing models cannot resist quantum attacks, which poses the risk of data leakage. The research on the flight data sharing model against quantum attack has become one of the research hotspots. Lattice-based cryptography is recognized as an effective way to resist quantum attacks. A flight data sharing model on consortium blockchain is proposed in this paper to resolve data leakage during data sharing. First, a new lattice-based multisignature scheme (Lamus) is proposed, capable of resisting quantum attacks. We prove the security of the proposed Lamus scheme in the random oracle model. Moreover, a flight data sharing model on consortium blockchain is proposed by applying the proposed Lamus scheme to resist quantum attacks. Security and performance analysis show that the model guarantees antiquantum security, and it achieves good performance in terms of storage efficiency and operating efficiency.

1. Introduction

The Air Transport industry is a highly interconnected industry that relies on data sharing to function properly. Data sharing process is vulnerable to attack such as eavesdropping attacks, which makes many data owners unwilling to share flight data. In view of the above reason, existing aviation data sharing systems can only realize data sharing in a certain range, which affects the effect of aviation data sharing to different degrees.

With the development of network technology, network security requirements are also increasing. Blockchain is a representative high-security network technology, which uses a unique data structure to verify and store data. Blockchain uses cryptography to achieve its nontampering. Digital signature technology is used to ensure the integrity and correctness of data transmission in blockchain.

The use of multisignature schemes in blockchain is mainly to improve storage efficiency by reducing the size of signatures, and most of these multisignature schemes are based on elliptic curves. With the development of quantum computers, the security of the existing signature schemes based on elliptic curves or integer factorization confronts severe challenges. Lattice-based cryptography is a typical postquantum cryptography [1]. This paper proposes a new lattice-based multisignature scheme and applies it to a blockchain-based flight data sharing system to improve the quantum resistance of the data sharing scheme.

This paper mainly consists of the following parts. The first part is a brief introduction to the background of this paper. The second part is an overview of related works. In the third part, a lattice-based multisignature scheme is proposed. The security and correctness of lattice-based multisignature schemes are proved in Section 4. The fifth part is the flight data sharing model based on the Lamus scheme. The last section concludes and lists some future work.

2. Related Work

In recent years, many schemes for flight data sharing systems have been proposed, such as Air Traffic Management data sharing. However, there are some cybersecurity threats in flight data sharing: eavesdropping and jamming. The interference in the data sharing process is mainly caused by malicious users publishing wrong information, which leads to wrong judgments on aircraft and ground stations. The authenticity and integrity of shared data is very important to management. In order to prevent the flight data from being maliciously tampered with, digital signature technology is used in many flight data sharing mechanisms. In [2], He et al. proposed a certificateless designated verifier proxy signature scheme to solve the problems of digital certificate management and key escrow in unmanned aerial vehicle networks. In [3], Li and Pu proposed a lightweight digital signature protocol to protect drones from man-in-the-middle attack. In [4], the authors proposed a lightweight authentication and key agreement scheme to implement the authentication problem between drones and users. In order to protect the integrity of flight data, a variety of digital signature schemes have been proposed. However, research of postquantum signature for flight data has not received much attention.

Blockchain is a decentralized distributed storage system, and the data stored in it is immutable. Many flight data sharing systems have been proposed on the basis of blockchain [5, 6]. In [7], the authors implemented secure communication between aircraft and ground station on the basis of blockchain. In [8], the authors implemented a blockchain-based PKI that implements "Secure Broadcast Authorization." In [9], the authors treated each flight plan as a blockchain transaction but does not deal with the verification of the flight data. Reference [10] proposed a blockchain and cloud storage based framework to guarantee the unmanned aerial vehicle data integrity. In [11], the authors presented a data integrity assurance mechanism based on digital signatures in IoT environment. Reference [12] proposed a data placement strategy named DE-DPSO-DPS to consider shared datasets across various geographically distributed environments.

An important concept in blockchain technology is accounts, which includes multiple signature accounts [13]. The multisignature was first proposed by Itakura, which refers to a group of signers signing the same binary string information and finally forming a signature result of the size like a single ordinary signature [14]. A multisignature scheme enables a group of signers to produce a compact, joint signature on a common document and has many potential uses [15]. Micali et al. were the first to give the formal model and security proof of multisignature and proposed a multisignature scheme based on the Schnorr signature scheme [16]. If multisignature is not applied, the size of each block in the blockchain will be much larger. Therefore, the combination of multisignature and blockchain received widespread attention. The widely used signature scheme in the blockchain is ECDSA digital based on the elliptic curve. Lindell et al. proposed a multiparty ECDSA signature scheme with distributed key generation. Kansal et al. proposed a lattice-based multisignature scheme for the first time at the 2020 African Cryptographic Conference and improved the original scheme in 2021 [17, 18]. Chen et al. proposed a novel certificateless multisignature scheme over NTRU lattices in [19].

2.1. Our Contribution.

This paper proposes a new identity-based multisignature scheme on lattice.

(1) A lattice-based multisignature scheme is proposed in this paper. In order to solve the high overhead of certificates storage and management in current signature schemes, a lattice-based multisignature scheme is proposed by introducing the identity cryptography mechanism into the multisignature schemes. Moreover, we prove the security of the proposed scheme in the random oracle model.

(2) A flight data sharing model on consortium blockchain is proposed in this paper. By applying the proposed lattice-based multisignature scheme into flight data management, a flight data sharing model on consortium blockchain is proposed in this paper. The proposed model solves data leakage during data sharing, flight resources wasting, and information leakage. The proposed model obtains high storage efficiency and high operating efficiency, ensuring high security against quantum attacks.

(3) The proposed flight data sharing model is secure against quantum attacks. The lattice-based multisignature scheme is secure under the assumption of the R–SIS hardness, which guarantees the postquantum security of our data sharing model.

3. Preliminary

3.1. Notations.

The notations used in this paper are presented in Table 1.

3.2. Blockchain.

Blockchain is a linked list data structure. Each block is linked to the previous one and stores a series of ordered things. It is a particular type of distributed database. The main difference between blockchain and ordinary databases is that blockchain guarantees decentralization, as well as the consistency, nontampering, and traceability of the stored data. Among them, the characteristic of decentralization is the most critical. Essentially, the blockchain can be seen as a decentralized database.

According to the degree of network centralization, blockchain is generally divided into public blockchain, private blockchain, and consortium blockchain. The public blockchain is completely decentralized. Users may join the blockchain at any time, participate in any activity in the blockchain, and maintain and manage the ledger. A private blockchain is a completely closed blockchain, and only private individuals participate in releasing blocks and storing blocks. The private blockchain usually runs in a relatively controllable and credible intranet environment. The consortium blockchain is a kind of blockchain for specific groups or organizations. It is a semipublic blockchain, and only certain members participate in releasing and storing blocks [20, 21].

Table 1: Notations.

Notations	Description
pp	System parameters
R, Z	Set of real numbers (integers)
n, q, l	Security parameter
R_q	$R_q = (R/qR) = (Z_q[x]/f(x))$
mpk, msk	Master public key and master private key
id, L_{id}	Identify signer and the list of signer private key
sk, L_{sk}	Private key of signer and the list of signer private key
pk	Aggregation public key
M	Message
σ	Multisignature of message
i	The order of signers, $1 < i < k$.
k	The number of participating signers

3.3. Basic Knowledge of Lattice Cryptography. Lattice-based cryptography is the use of conjectured hard problems on point lattices in R^n as the foundation for secure cryptographic systems. Lattice-based cryptography is a typical postquantum cryptography [22, 23]. Another attractive feature of it is security under worst-case intractability assumptions [24].

Definition 1. (**Lattice**)

If $B = (b_1, b_2, \ldots, b_m)$ by R^n is composed of m linearly independent vectors, then the lattice $L(B)$ is defined as the linear combination of all integer coefficients of this group of vectors, denoted as

$$\begin{aligned}\Lambda &= L(B)\\ &= \left\{\sum_{i=1}^{m} x_i b_i | x_i \in Z\right\}.\end{aligned} \quad (1)$$

Let n be the dimension of the lattice $L(B)$, m is the rank, and B is a set of bases of the lattice.

Definition 2. ($\mathbf{R\text{-}SIS_{n,m,q,\beta}}$ problem)

Given a security parameter n and a uniform random matrix $A = [a_1, a_2, \ldots, a_m] \in R_q^{1\times m}$, find a nonzero solution $u = (u_1, u_2, \ldots, u_m)^T \in R^m$ that satisfies the following conditions:

$$Au = 0 \bmod q, \quad \|u\|_\infty \le \beta. \quad (2)$$

3.4. iNTRU Trapdoor. The iNTRU is an inhomogeneous NTRU problem [25]. This paper mainly uses a kind of gadget-based iNTRU lattice trapdoor. Compared with other trapdoor functions, the size of the signature and key generated by the iNTRU trapdoor has obvious advantages. At the same time, the calculation process has no obvious disadvantage compared with other trapdoor generating functions, so we choose the iNTRU trapdoor function [26]. The iNTRU lattice trapdoor function contains two algorithms, which, respectively, implement the functions of trapdoor generation and preimage sampling (see Algorithm 1 and Algorithm 2).

3.5. Lattice-Based Multisignature Scheme (Lamus). The lattice-based multisignature scheme proposed in this paper is postquantum security under the assumption of the R–SIS hardness. At the same time, our scheme is an identity-based multisignature scheme, which directly uses the signer's identity ID as the public key. Identity-based signature solves the high overhead of certificate storage and management in PKI-based signature schemes. In addition, our scheme uses the TrapGen and the Sample in the iNTRU scheme to improve the efficiency of the key extraction algorithm. In this paper, we prove the security of the scheme in the random oracle model.

There are several types of entities in the scheme: Key Generation Center (KGC), signers, designated signer, and verifier. KGC is responsible for the production of public parameters and the signer's private key in the multisignature scheme. The signer and the designated signer jointly generate a multisignature to protect the integrity of the signed data. The verifier verifies the multisignature to confirm its integrity.

The lattice-based multisignature scheme includes five algorithms: ParameterGen, ExtractKey, Aggregation, Sign, and Verify. The ParameterGen algorithm is run by KGC to generate system parameters and the master key pair of KGC. The system parameter pp is the input of all other algorithms in the scheme. The Extract Key algorithm is run by the KGC and generates signer's private key after receiving the identity of the signer. The Aggregation algorithm is usually run by a designated signer. The designated signer generates an aggregated public key from the IDs of all signers to verify the validity of the signature. (The Aggregation algorithm can also be run by the verifier.) The Sign algorithm is run by multiple signers and the designated signer. The signers generate their own partial signatures and send them to the designated signer. The designated signer generates a single multisignature from multiple partial signatures. The final output of the Sign algorithm is a multisignature. The Verify algorithm is run by the verifier and outputs 0 or 1 as the verification result. The overall structure is shown in Figure 1.

KGC runs the system parameter generation function. This function generates related parameters of multisignature, such as master key pairs. The specific algorithm steps are described in Algorithm 3.

When receiving the identity id_i sent by the signer, KGC generates the corresponding private key sk_i for the signer (see Algorithm 4).

The aggregation public key algorithm (Lamus.Aggregation) is executed by the specified signer or verifier. L_{id} is the identity list of signers (see Algorithm 5).

M is the message for all signers to sign. L_{sk} is a list of all signers' private keys. The signing process is divided into three steps. The first step is mutual interactions between signers to negotiate unanimous signature parameter R. The signers generate their partial signatures Z_i using their private key in the second step. In the third step, all signers send their partial signatures to the designated signer, and the designated signer combines the received partial signatures to form a multisignature σ (see Algorithm 6).

```
(i) **Input:** The security parameter $1^{\lambda}$.
(ii) **Output:** A function-approximate trapdoor partial $(vk, T)$.
(1) Sample $r \leftarrow X, e \leftarrow X^{m}$.
(2) Set $A: = r^{-1}(f + e) \in R_q^n$
(3) **Return** $(vk, T): = (A, (r, e)) \in R_q^{n+1} \times R_q^{n+1}$
```

Algorithm 1: iNTRU.TrapGen.

```
(i) **Input:** A cost $u \in R_q$, and a trapdoor $(r, e)$.
(ii) **Output:** An approximate pre-image $u$.
(1) Sample perturbation $p \leftarrow D_{R^{n+1}}, \sqrt{\sum} \leq$ for $\sum p: = s^2 I_{n+1} - \sigma_g^2 \begin{bmatrix} e^t d & -\bar{r}e^t \\ -re & r\bar{r}I_m \end{bmatrix}$.
(2) Set coset $v: = u - \mathrm{A}^{\mathrm{t}}\mathrm{p} \in \mathrm{R}_{\mathrm{q}}$.
(3) Sample the G-lattice $x = (x_1, x_2) \leftarrow \mathrm{SamplePre}(v, \sigma_g)$.
(4) Define the $i$NTRU $\cdot$ trapdoor as $R: = \begin{bmatrix} -e^t \\ rI_m \end{bmatrix}$.
(5) Set the approximate pre-image $y: = Rx_2 + p \in R^m$.
(6) **Return** $y$.
```

Algorithm 2: iNTRU.SamplePre.

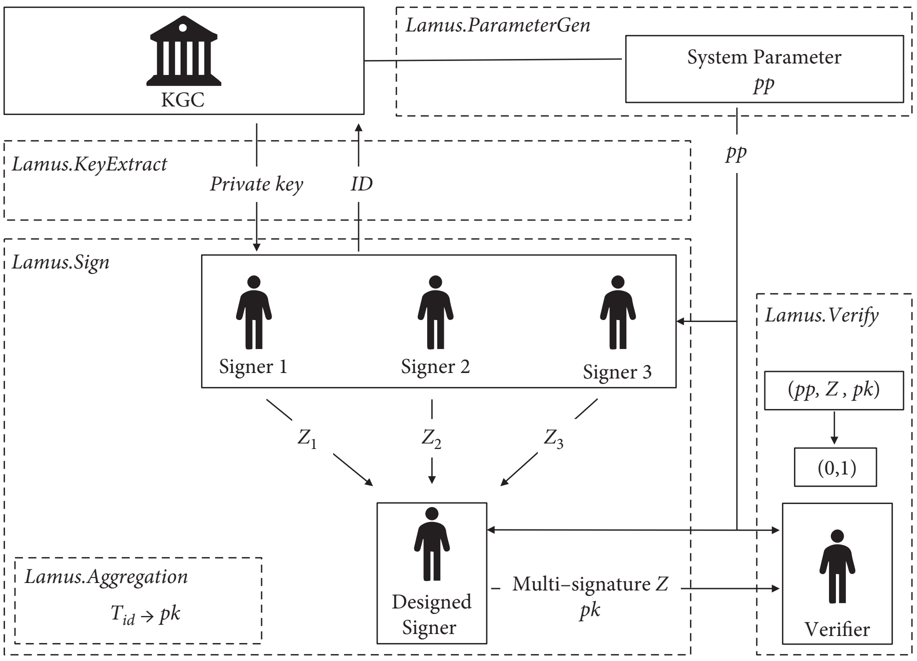

Figure 1: The lattice-based multisignature scheme.

The verifier extracts the M, Z, pk, r from the multi-signature and verifies the validity of the received multi-signature (see Algorithm 7).

3.6. *Security Analysis.*

This section contains the correctness and security proofs of the multisignature scheme.

3.6.1. *Proof of Correctness.*

A multiple signature generated according to the above scheme can be successfully verified, which means that the scheme proposed in this paper is correct. If signers correctly generate a multisignature, then the verifier can verify the following equation using A, Z, R in the signature, the result of the hash function c, and the aggregation public key pk of signers.

```
(i) Input: The security parameter $1^\lambda$.
(ii) Output: The system parameter $pp$.
(1) Run the TrapGen function, input the security parameter $1^\lambda$, and get the system parameter $A \in R_q^n$ and the corresponding trapdoor $(r, e)$.
(2) Samples three cryptographically secure hash functions $H_0$: $R_q \longrightarrow R_q$ , $H_1$: $\{0,1\}^* \longrightarrow P^n$, $H_2$: $\{0,1\}^* \longrightarrow D_{32}^n$. $n, q$ is the parameter of $R_q^n$
(3) Publish system parament $pp = \{A, n, q, H_0, H_1, H_2\}$. Let $A$ be the master public key $mpk$ and $(r, e)$ the master private key $msk$.
```

Algorithm 3: Lamus.ParameterGen.

```
(i) Input: The system parameter $p$ , master key pairs $msk, mpk$, and the $i\ d$ of the signer.
(ii) Output: The private key $sk_i$ for the signer.
(1) Run the function SamplePre, input the identity $id_i$ and trapdoor $(r, e)$. Output the approximate pre-image $sk_i$ of $id_i$.
(2) Send the private key $sk_i$ to the signer via secure communication.
(3) Return 0.
```

Algorithm 4: Lamus.ExtractKey.

```
(i) Input: The system parameter $pp$, and the identity list $L_{id}$ of the signers.
(ii) Output: The aggregation public key $pk$ for the signers.
(1) Input the public key list $L_{id}$.
(2) Calculate $c_i' = H_1(L_{id}, id_i)$, for $1 < i < k$, $k$ is the number of signers in the identity list.
(3) Computer $pk = \sum_i^k id_i \cdot c_i'$, for $1 < i < k$.
```

Algorithm 5: Lamus.Aggregation.

```
(i) Input: The system parameter $pp$, and the private key list $L_{sk}$ of the signers.
(ii) Output: The multisignature $\sigma = (Z, pk, M, R)$.
(1) All signers randomly select a vector $r_i$, calculate $v_i = A \cdot r_i$ and $t_i = H(v_i)$, and send $(t_i, v_i)$ to other signers.
(2) Computer $R = \sum_i^k v_i$, for $1 < i < k$, $k$ is the number of signers in the identity list. $c = H_2(M, R)$, and $c_i' = H_1(L_{id}, id_i)$, the $c_i = c \cdot c_i'$.
    The partial signature $Z_i = sk_i \cdot c_i + r_i$ is using the signer's private key $sk$.
(3) All signers send their own $Z_i$ and $r$ to the designated signer. The designated signer computers the signature $Z = \sum_i^k Z_i$, and runs the Lamus.Aggregation outputting the aggregate public key $pk$.
```

Algorithm 6: Lamus.Sign.

```
(i) Input: The system parameter $pp$, multisignature $\sigma$.
(ii) Output: 0 or 1.
(1) The verifier confirms the $Z \in R_q^n$, computers $c = H_2(M, R)$.
(2) The verifier verifies $A \cdot Z = c \cdot pk + R$. If the equation is true, output 1, otherwise output 0.
```

Algorithm 7: Lamus.Verify.

$$\begin{aligned} A \cdot Z &= A \cdot \sum_{i}^{k} z_i \\ &= A \cdot \sum_{i}^{k} sk_i \cdot c_i + r_i \\ &= \sum_{i}^{k} a \cdot sk_i \cdot c_i + \sum_{i}^{k} a \cdot r_i \\ &= \sum_{i}^{k} id_i \cdot c_i + \sum_{i}^{k} v_i \\ &= c \cdot \sum_{i}^{K} id_i \cdot c_i' + R \\ &= c \cdot \mathbf{pk} + R. \end{aligned} \tag{3}$$

The verifier can legally obtain the parameters of the left and the last row on right of equation (3). The correctness of the Lamus scheme is proved.

3.6.2. Proof of Security. The security of the proposed scheme mainly proves the unforgeability of multisignature. To prove the unforgeability of multisignature, we will prove the following theorem in the random oracle model.

Theorem 1. *When there exists at least one honest signer, our scheme satisfies the unforgeability in the random oracle model.*

Proof. of Theorem 1 In Theorem 2, we construct a security game between the simulator B and the malicious adversary A. Simulator B acts as KGC responds to adversary A queries. Meanwhile, simulator B controls the identity of an honest signer. Adversary A can act as all the signers except the honest signer. In the security game, it is assumed that the adversary A can forge a multisignature involving one honest signer. The simulator B can then use this ability of adversary A to solve the R–SIS problem instance. This is inconsistent with the fact that R–SIS does not have a polynomial-time solution algorithm, which proves the assumptions wrong in the security game. If Theorem 2 holds, it is proved that the multiple signature scheme with an honest signer is secure. Theorem 2 holds; then Theorem 1 holds. □

Theorem 2. *Suppose there exists a polynomial-time adversary A, who makes at most h_0, h_1, h_2 queries to H_0, H_1, H_2 oracle, makes s queries to signature oracle, and succeeds in forging a multisignature with nonnegligible advantage ϵ within time t. Then there exists a simulator B that uses the ability of adversary A to solve $R - SIS_{(q,n,m,d)}$ problem instance within time $t + t_1 + t_2$ (t_1 is the running time of TrapGen, t_2 is the running time of SamplePre with the advantage of (ϵ/q_H), $q_H = h_0 + h_1 + h_2$).*

Proof. of Theorem 2 The proof of the scheme is proved by a security game and analysis of game output. The security game includes two entities, simulator B and adversary A. The security game has three stages: Setup, Oracle query, Forgery. If the multisignature is successfully forged, the game is over. We analyse the output of the security game to get a solution to the $R - \mathrm{SIS}_{(q,n,m,d)}$ problem. □

3.7. Security Game

3.7.1. Setup. Simulator B runs the function TrapGen and obtains $(A, (r, e))$. Simulator uses A and (r, e) as the master key pairs and randomly extracts an id^* from the identity list L_{id} as the identity of the honest signer and runs *SamplePre* to extract the original image sk^* of id^* as the private key.

3.7.2. Oracle Query. The adversary A carries out ExtractKey oracle, H_1 oracle, H_2 oracle, and signature oracle queries.

(1) ExtractKey Oracle Query. First, simulator B checks whether the queried id is stored in the ExtractKey oracle query table. If it is in the table, return the result directly to the adversary A. Else if the queried id (not id_{i^*}) exists in the identity list L_{id}, simulator B returns the corresponding private key sk for the adversary A and stores it. If the queried id does not exist in the identity list L_{id} or the queried $id = id_{i^*}$, abort.

(2) $\mathbf{H}_1$ *Oracle Query.* Simulator B checks whether the queried (id_i, L_{id}) is stored in the H_1 oracle query table. If it is in the table, return the result directly to the adversary A. Else if the queried id_i exists in the identity list L_{id}, simulator B returns the corresponding $\mathrm{Q_I}$ from the Q_{q_H} for the adversary A and stores it. If the queried id does not exist in the identity list L_{id}, randomly select a result to return.

(3) $\mathbf{H}_2$ *Oracle Query.* Simulator B checks whether the queried (M_l, r_l) is stored in the H_2 oracle query table. If it is in the table, return the result directly to the adversary A. Else simulator B returns the corresponding $H_2(M_l, r_l)$ for the adversary A and stores it.

(4) Signature Oracle Query. Firstly, simulator B checks whether the (id_{i^*}) is existing in the queried identity list L'_{id}. If it is not in the L'_{id}, abort. Else simulator B checks whether the queried (M_l) is stored in the H_2 oracle query table. Then simulator B returns the corresponding σ_{i^*} for the adversary A and stores it.

3.7.3. Forgery. The adversary A produces a forged result of the message M; if the L_{id} corresponding to the message does not contain id_{i^*}, abort. And message M should be queried by H_2 oracle or signature oracle. Check the validity of the signature. If the signature is valid, the security game is over.

3.8. Analysis of Security Game Output. By using T_{id}, simulator B obtains T_{sk}. Simulator B makes another signature Z' by using T_{sk}. Because the random number r_{i^*} corresponding to the signature is randomly selected by the simulator B, we make R of the two signatures the same. According to this conclusion, we obtain the following equation:

$$\begin{aligned} R &= R', \\ AZ - pk \cdot c &= AZ' - pk \cdot c, \\ A(Z - Z') &= 0, \\ A(T_{sk} \cdot c - T'_{sk} \cdot c) &= 0, \\ A\left[\left(sk_{i^*} - sk_{i^{*\prime}}\right) \cdot c\right] &= 0. \end{aligned} \tag{4}$$

Then the simulator B finds the solution $Q = (sk_{i^*} - sk_{i^{*\prime}}) \cdot c$ of a $R - \mathrm{SIS}_{(q,n,m,d)}$ problem instance, and Q satisfies $A \cdot Q = 0$. Theorem 2 is proved.

4. Lamus-Based Flight Data Sharing Model on Consortium Blockchain

The current traditional flight data sharing model has privacy leakage in the quantum computer environment. This section proposes a flight data sharing model on consortium blockchain. The proposed flight data sharing model realizes flight data sharing among different agencies. At the same time, this model uses the lattice-based multisignature scheme proposed in the third section, enhancing security against quantum attacks. Our flight data sharing model ensures data integrity when data is shared between different institutions. In addition, the unforgeability of flight data and the traceability of signer are obtained.

4.1. Model Architecture. The model consists of four types of entities: Flight Data Center (FDC), Flight Management Agency (FMA), KGC, and client user (client). The clients include aircrew, airline, ground personnel, and data consumers. Client users do not participate in the operation and storing of the blockchain. Aircrew, airline, and ground personnel sign the flight data together and submit it to FDC. The FDC is the main component of the blockchain. It is responsible for storing blocks and providing transaction uploads and transaction queries for clients. The FMA is responsible for the ordering of transactions and the releasing blocks. The KGC is responsible for providing the private key query for the client user. The user provides the KGC with an ID, and the KGC sends the user's private key to the user through a secure channel.

The generation of the transaction requires the participation of aircrew, airline, and ground personnel. FDC forwards the transaction sent by the client to FMA. The following is the process of block release. Transactions sent to the FMA are deposited in the transaction pool in order. When the number of transactions is enough, the transactions are packaged into blocks, which are sent to all FDCs. After receiving the blocks, the FDC updates the local blockchain. In the transaction inquiry process, the data consumer first initiates a flight data request to the FDC. After receiving the request, the FDC performs signature verification on the request message. If the received request is valid, FDC extracts the flight data from the corresponding transaction. The FDC sends the flight data to the data consumers.

The data of the blockchain data sharing system is stored in blocks. The block is mainly divided into two parts: the Block Header and the Block Body. The Block Header stores PreHash, MerkleRoot, TimeStamp, and Index. PreHash stores the hash result of the previous block to ensure data immutability. The MerkleRoot in the Block Header is generated from the hash values of all transactions in the Block Body. Through MerkleRoot, it is possible to quickly determine whether a transaction is included in this block. TimeStamp marks the release time of the block. Index is the height of the block in the blockchain. The Block Body stores multiple transactions. The flight data and corresponding multisignatures are stored in the transaction.

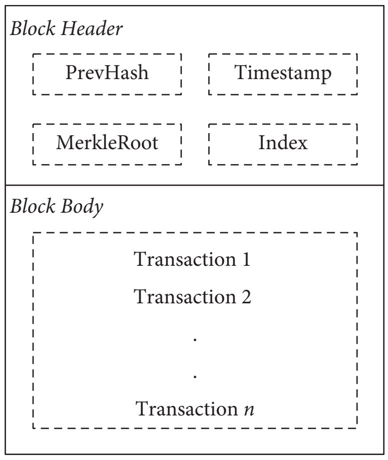

Figure 2: The structure of block.

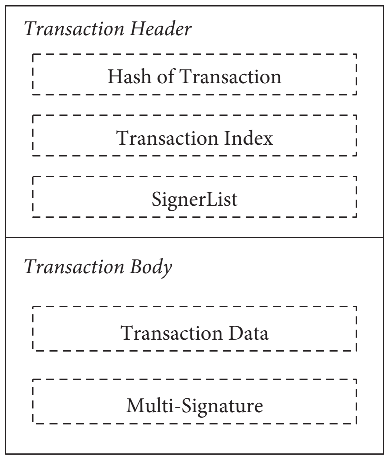

Figure 3: The structure of transaction.

The structure of transaction and block is shown in Figures 2 and 3.

Because client users do not participate in specific transactions and block storage, the model provides three interfaces for clients.

4.1.1. Key Registration Interface. The reply from this interface is answered by KGC, and the client users obtain the private key that matches their identity through this interface.

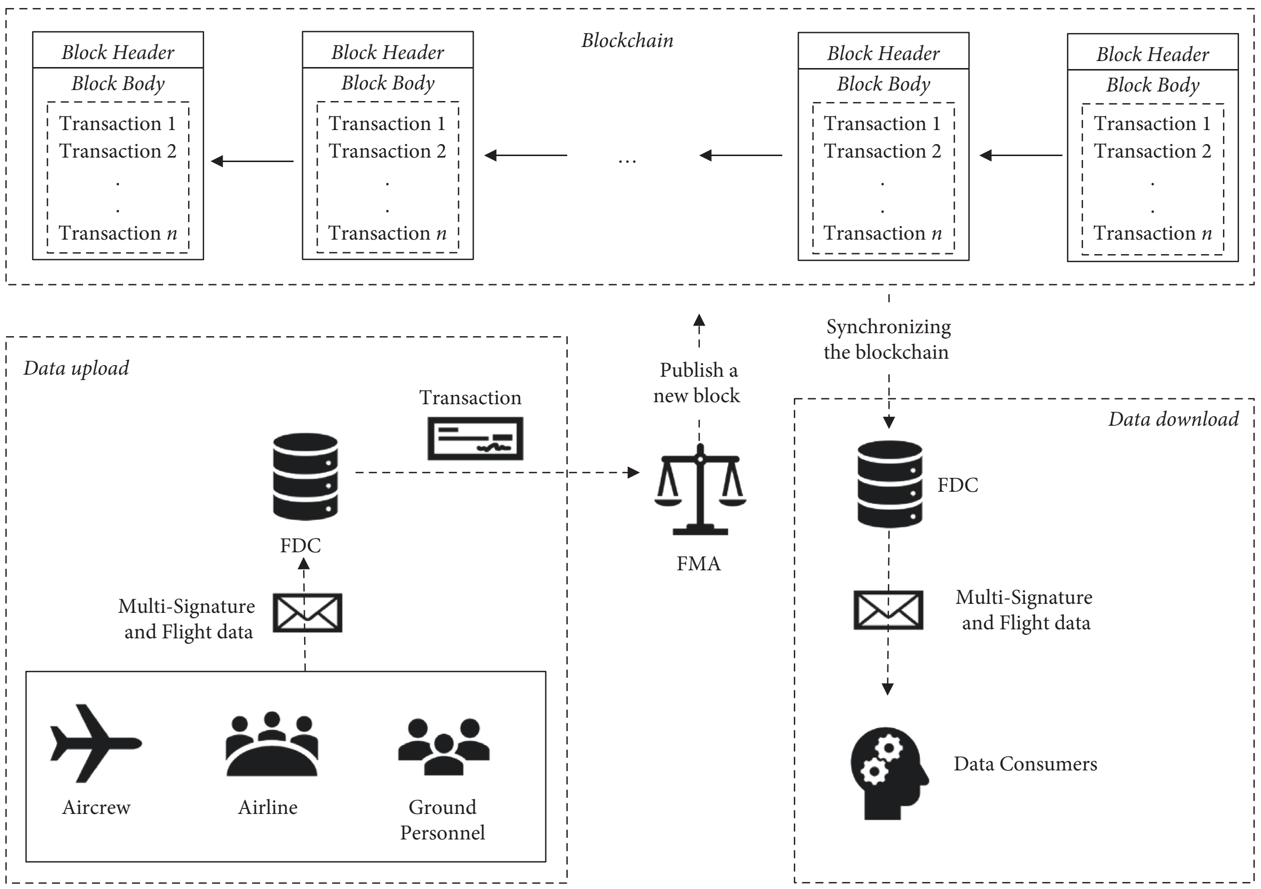

Figure 4: Model architecture.

4.1.2. Transaction Generation Interface. This interface is mainly used by FDC to accept flight data and multisignature uploaded by client users. After receiving the relevant data, FDC generates a transaction and forwards it to FMA.

4.1.3. Transaction Query Interface. The reply from this interface is answered by FDC. Client users apply to the FDC to query a transaction on the blockchain through this interface.

Figure 4 contains "data upload" and "data download" areas. The area on the left shows the process of FDC submitting a transaction. The area on the right shows the process of querying a transaction. Aircrew, airline, and ground personnel can perform these two operations, but the only one-way operation is shown in the figure.

In this model, all clients need to obtain the key corresponding to the identity from the KGC through the key registration interface. The detailed process is as follows.

(1) KGC runs $TrapGen$, inputs security parameter 1^λ, and outputs trapdoor pair $[A, (r, e)] (A \in \mathcal{R}_q^n)$.

(2) Randomly select H_0: $\mathcal{R}_q \longrightarrow \mathcal{R}_q$, H_1: $\{0,1\}^* \longrightarrow P^n$, H_2: $\{0,1\}^* \longrightarrow D_{32}^n$. n, q is the parameter of $\mathcal{R}_q^n$.

(3) Public system parameters $pp = \{A, n, q, H_0, H_1, H_2\}$. Let A be master public key mpk; (r, e) is master private key msk.

(4) The signer sends the identity to the KGC. KGC runs $SamplePre$ after receiving the identity and inputs the signer identity id_i and master private key (r, e). KGC outputs the original image sk_i of id_i.

(5) KGC sends the signer's private key sk_i to the signer through a secure channel.

4.2. Sharing of Flight Data. The model is divided into three stages: the creation of the transactions, block release, and transaction query. The block creation in the data sharing system is the process of data upload. Data is stored in transactions of block and uploaded as the block is released. The transaction query is corresponding to the data download in the data sharing system. Data consumers obtain the flight data through the corresponding transaction.

4.3. The Creation of the Transactions. To prevent malicious users from creating false flight data, an effective flight data needs to be signed jointly by aircrew, airline, and ground personnel.

The transaction generation process is as follows:

(1) Aircrew, airline, and ground personnel generate their partial signatures of flight data. They sign Z_i for the data M generated during the flight process.

(2) The FDC composes the partial signatures of all signers into multisignature and generates a transaction with multisignature and flight data. As a designed signer, the FDC generates an aggregate

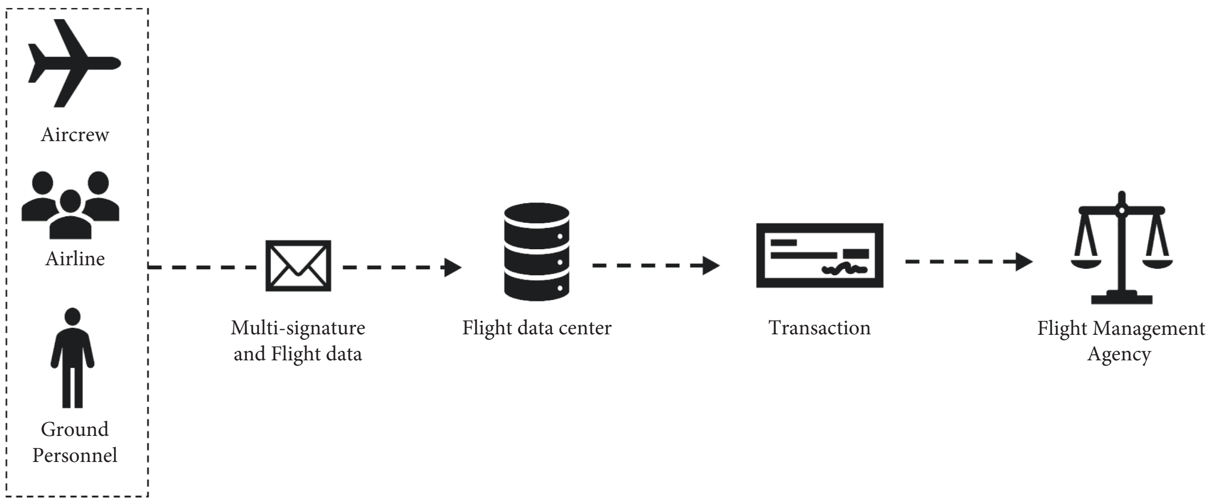

Figure 5: Creation of transactions.

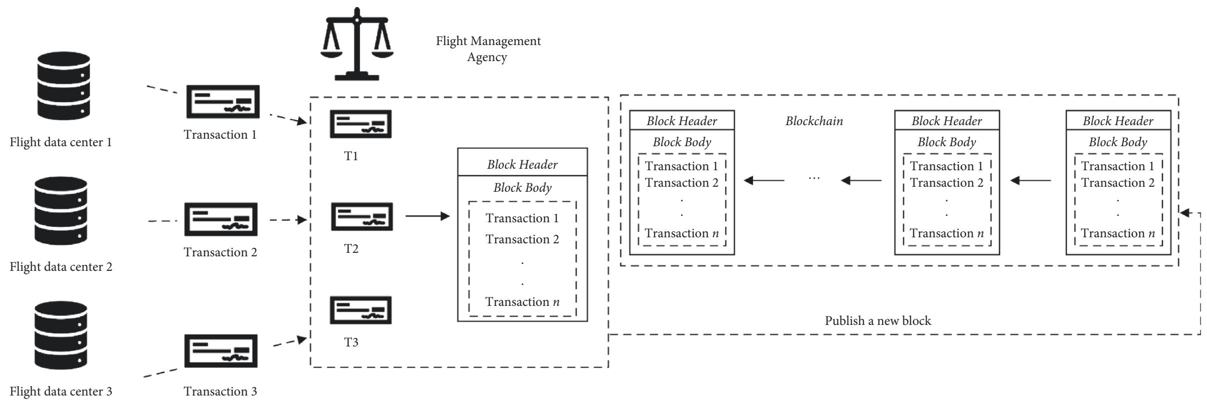

Figure 6: Block release.

public key and multisignature σ. FDC generates a transaction T by using the message M, σ, and **pk**.

(3) Furthermore, the transaction T is submitted to the FMA by the FDC.

The process of transaction generation is contained in Figure 5. The specific process is as follows:

(1) First, the aircrew, airline, and ground personnel, respectively, generate a random polynomial r_i, calculate $v_i = A \cdot r_i$ and $t_i = H(v_i)$, and send (t_i, v_i) to other signers.

(2) Calculate $R = \sum_i^k v_i$, for $1 < i < k$. Calculate $c = H_2(M, R)$, $c_i' = H_1(L_{id}, id_i)$, $c_i = c \cdot c_i'$. Aircrew, airline, and ground personnel use sk to generate partial signatures $Z_i = sk_i \cdot c_i + r_i$.

(3) All signers send partial signatures Z_i and r to the FDC. The FDC calculates $Z = \sum_i^k Z_i$ and the aggregate public key $pk = \sum_i^k id_i \cdot c_i'$. The FDC uses multiple signatures and flight data to generate a transaction T_i.

(4) The FDC sends the generated transaction T_i to the FMA.

4.4. Block Release. The FMA sorts the received transaction information to form a transaction pool $(T_1, T_2, T_3, \ldots, T_n)$. The FMA takes out the transactions in the transaction pool to generate blocks $\{B_1 = (T_1, T_2, T_3, \ldots, T_j)\}$ and sends the generated blocks to each FDC. Figure 6 shows the process of block release.

The specific process is as follows:

(1) The FMA extracts a set of (Z, pk, m, R) from the transaction T_i and calculates $c = H_2(m, R)$.

(2) The FMA verifies the following equation, $A \cdot Z = c \cdot pk + R$. If the equation holds, the transaction is considered legal.

(3) Add legitimate transactions to the transaction pool.

(4) The FMA writes the timestamp, the Merck tree, the current block ID, and the hash of the previous block into the Block Header. Write the transaction to the block.

(5) The FMA broadcasts the block to all FDCs.

4.5. Transaction Query. Transaction queries are initiated by data consumers to the FDC. The FDC verifies the request message after receiving the request. If the message is

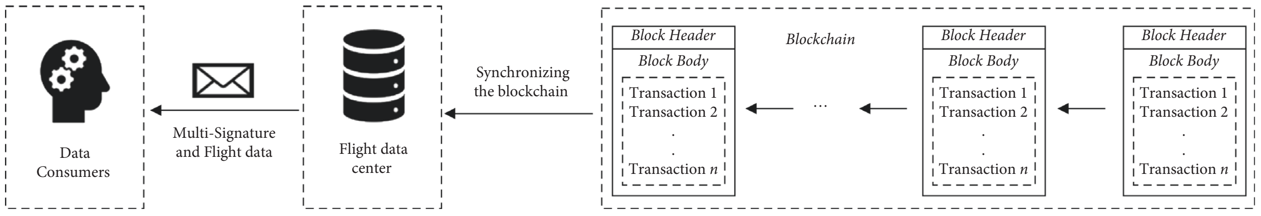

Figure 7: Transaction query.

successfully verified, the transaction in the blockchain is queried. If the verification fails, an error will be returned. After querying the transaction information, the FDC needs to extract the flight data and multiple signatures in the transaction information. The FDC checks whether the requester is in the multisignature public key list. Furthermore, the FDC returns the corresponding flight data of the queried transaction to the requesting user if the requester's identity is in the public key list. Figure 7 shows the process of transaction query.

The detailed process is as follows:

(1) Data consumers initiate a request for flight data. The request message contains the message itself and the multisignature of the message.

(2) The FDC extracts a set of (Z, pk, m, R) from the request message and calculates $c = H_{2(m,R)}$.

(3) The FDC verifies the following equation, $A \cdot Z = c \cdot pk + R$. If the equation holds, the request message is considered legal. Otherwise, FDC returns an error message to the querier.

(4) FDC performs message query in blockchain and returns error messages if message query fails. After the query is successful, FDC returns the query data. The FDC returns the flight data to the querier.

5. Conclusion

This paper proposes a lattice-based multisignature scheme (Lamus) and applies it to the flight data sharing model on consortium blockchain. The proposed data sharing model has solved the problem of data islands and the information leakage of flight data. In addition, the proposed Lamus scheme is secure against quantum attacks.

The proposed Lamus scheme requires the interaction of signers. In the scenario of the proposed data sharing model, the efficiency of Lamus scheme achieves the desired effect. However, the network delay affects the efficiency of the Lamus scheme when the signers are in different locations. Therefore, the efficiency of the Lamus scheme is affected by network conditions. In future work, we will consider constructing a noninteractive multisignature scheme to avoid network conditions affecting its efficiency.

Data Availability

No data were used to support this study.

Disclosure

An earlier version has previously been published as conference [27].

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This study was supported by Natural Science Foundation of Shandong Province, no. ZR2019MF062.

References

[1] A. Deo, L. Beno, K. Nguyen, and O. Sanders, "Lattice-based E-cash, revisited," in *Advances in Cryptology – ASIACRYPT*Springer International Publishing, Daejeon, Korea, 2020.

[2] L. He, J. Ma, L. Shen, and D. Wei, "Certificateless designated verifier proxy signature scheme for unmanned aerial vehicle networks," *Science China Information Sciences*, vol. 64, no. 1, Article ID 112101, 2020.

[3] Y. Li, *DroneSig: Lightweight Digital Signature Protocol for Micro Aerial Vehicles*, Marshall University, Huntington, West Virginia, 2020.

[4] Y. Zhang, D. He, L. Li, and B. Chen, "A lightweight authentication and key agreement scheme for Internet of Drones," *Computer Communications*, vol. 154, pp. 455–464, 2020.

[5] G. D'Angelo, S. Ferretti, and M. Marzolla, "A blockchain-based flight data recorder for cloud accountability," in *Proceedings of the 1st Workshop on Cryptocurrencies and Blockchains for Distributed Systems*, Munich, Germany, June 2018.

[6] D. Cocîrlea, G. Jeon, M. M. Hassan, M. R. Hassan, and K. Kaur, "Blockchain in intelligent transportation systems," *Electronics*, vol. 9, no. 10, p. 1682, 2020.

[7] A. Arora and S. K. Yadav, "BATMAN: blockchain-based aircraft transmission mobile ad hoc network," in *Proceedings of the 2nd International Conference on Communication, Computing and Networking*, March 2019.

[8] R. Reisman, "Air Traffic Management Blockchain Infrastructure for Security," in *Proceedings of the AIAA Scitech 2019 Forum, American Institute of Aeronautics and Astronautics*, San Diego, CA, USA, January 2019.

[9] I. S. Bonomo, I. R. Barbosa, L. Monteiro et al., "Development of SWIM registry for Air traffic management with the blockchain support," in *Proceedings of the 2018 21st*

International Conference on Intelligent Transportation Systems (ITSC), Maui, HI, USA, November 2018.

[10] X. Liang, J. Zhao, S. Shetty, and D. Li, "Towards data assurance and resilience in IoT using blockchain," in *Proceedings of the MILCOM 2017 - 2017 IEEE Military Communications Conference (MILCOM)*, Baltimore, MD, USA, October 2017.

[11] N. Kammoun, A. ben Chehida Douss, R. Abassi, and S. Guemara el Fatmi, *Ensuring Data Integrity Using Digital Signature in an IoT Environment*, Springer International Publishing, Manhattan, NY, USA, 2022.

[12] X. Du, S. Tang, Z. Lu, J. Wet, K. Gai, and P. C. K. Hung, "A novel data placement strategy for data-sharing scientific workflows in heterogeneous edge-cloud computing environments," in *Proceedings of the 2020 IEEE International Conference on Web Services (ICWS)*, Beijing, China, October 2020.

[13] G. Maxwell, A. Poelstra, Y. Seurin, and W. Pieter, "Simple Schnorr Multi-Signatures with Applications to Bitcoin," *Designs, Codes and Cryptography*, vol. 87, pp. 1–26, 2018.

[14] K. Itakura and K. Nakamura, "A public-key cryptosystem suitable for digital multisignatures," *NEC Research \& Development*, vol. 71, pp. 1–8, 1983.

[15] M. Bellare and G. Neven, "Multi-signatures in the plain Public-Key Model and a General Forking Lemma," in *Computer Science, Mathematics '06*ACM, Alexandria, Virginia, USA, 2006.

[16] S. Micali, K. Ohta, and L. Reyzin, "Accountable-subgroup multisignatures," in *Proceedings of the 8th ACM Conference on Computer and Communications Security*, November 2001.

[17] M. Kansal, A. K. Singh, and R. Dutta, "Efficient multi-signature scheme using lattice," *The Computer Journal*, no. bxab077, 2021.

[18] M. Kansal and R. Dutta, "Round optimal secure multi-signature schemes from lattice with public key aggregation and signature compression," in *Proceedings of the International Conference on Cryptology in Africa*, July 2020.

[19] X. Chen, Q. Huang, and J. Huang, "A Novel Certificateless Multi-Signature Scheme over NTRU Lattices," in *Proceedings of the International Conference on Information Security Practice and Experience*, December 2021.

[20] E. Androulaki, A. Barger, V Bortnikov et al., "Hyperledger fabric: a distributed operating system for permissioned blockchains," in *Proceedings of the Thirteenth EuroSys Conference*, Association for Computing Machinery, Porto Portugal, April 2018.

[21] T. Li, Y. Chen, Y. Wang et al., "Rational protocols and attacks in blockchain system," *Security and Communication Networks*, vol. 2020, Article ID 8839047, 11 pages, 2020.

[22] C. Peikert, "A Decade of Lattice Cryptography," *Foundations and Trends® in Theoretical Computer Science*, vol. 10, 2015.

[23] V. Lyubashevsky, N. K. Nguyen, and G. Seiler, "SMILE: set membership from ideal lattices with applications to ring signatures and confidential transactions," in *Advances in Cryptology – CRYPTO 2021*Springer International Publishing, Manhattan, NY, USA, 2021.

[24] C. P. Schnorr, "Efficient signature generation by smart cards," *Journal of Cryptology*, vol. 4, no. 3, pp. 161–174, 1991.

[25] N. Genise, C. Gentry, H. Shai, and B. Li, "Homomorphic encryption for finite automata," in *Advances in Cryptology (ASIACRYPT 2019)*, Springer International Publishing, Cham, Switzerland, 2019.

[26] N. Genise and B. Li, "Gadget-based iNTRU lattice trapdoors," in *Proceedings of the International Conference on Cryptology in India*, December 2020.

[27] Y. Cui, S. Yu, and F. Li, *A Lattice-Based Anonymous Authentication for Privacy Protection of Medical Data*, Springer Singapore, Singapore, 2021.