Computers, Materials & Continua

Tech Science Press

Doi:10.32604/cmc.2025.073837

ARTICLE

Design of Consensus Algorithm for UAV Swarm Identity Authentication Based on Lightweight Blockchain

Yuji Sang[1], **Lijun Liu**[1,*], **Long Lv**[1,*], **Husheng Wu**[2] and **Hemin Yin**[1]

[1]Key Laboratory of Information and Network Security, Engineering University of the PAP, Xi'an, 710086, China
[2]The School of Equipment Management and Support, Engineering University of PAP, Xi'an, 710086, China

*Corresponding Authors: Lijun Liu. Email: cainan2001@163.com; Long Lv. Email: wjgcdianzijishu@163.com

Received: 26 September 2025; Accepted: 11 December 2025; Published: 12 March 2026

ABSTRACT: Aiming at the challenges of low throughput, excessive consensus latency and high communication complexity in the Practical Byzantine Fault Tolerance (PBFT) algorithm in blockchain networks, its application in identity verification for distributed networking of a drone cluster is limited. Therefore, a lightweight blockchain-based identity authentication model for UAV swarms is designed, and a Credit-score and Grouping-mechanism Practical Byzantine Fault Tolerance (CG-PBFT) algorithm is proposed. CG-PBFT introduces a reputation score evaluation mechanism, classifies the reputation levels of nodes in the network, and optimizes the consensus process based on grouping consensus and BLS aggregate signature technology. Experimental results demonstrate that under identical experimental conditions, compared with the PBFT algorithm, CG-PBFT achieves a 250% increase in average throughput, a 70% reduction in average latency, and simultaneous enhancement in security, thus making it more suitable for UAV swarm networks.

KEYWORDS: UAV swarm network; blockchain; PBFT consensus algorithm; credit score

1 Introduction

In recent years, research on unmanned aerial vehicle (UAV) swarms has developed rapidly, especially in the military field, where they have demonstrated significant effectiveness in performing tasks such as strikes, reconnaissance, and fire support. In these tasks, continuous data exchange is required between UAVs and between UAVs and the Ground Control Station (GCS). To prevent UAVs from being captured by the enemy during missions and ensure the confidentiality and security of communications, identity authentication is crucial as the primary line of defense. However, traditional identity authentication methods struggle to balance security and efficiency. Blockchain, as an emerging distributed infrastructure, adopts a block-chain data structure. Through encryption algorithms and consensus mechanisms, it stores data in a chained form of blocks, ensuring the characteristics of data such as immutability [1], transparency [2], and decentralization [3]. It also enables data editing and operations with the help of smart contracts. Leveraging these inherent advantages, blockchain technology offers a viable solution to address the security challenges confronting UAV swarms. As the core of blockchain technology, the key role of the consensus algorithm lies in ensuring that all nodes in the distributed network reach an agreement on the state of the blockchain, while maintaining the security, integrity, and immutability of data [4].

Currently, prevalent blockchain consensus mechanisms primarily encompass Proof of Stake (PoS) [5,6], Proof of Work (PoW) [7,8] and Practical Byzantine Fault Tolerance (PBFT) [9], etc. The PBFT consensus mechanism originates from the Byzantine Generals Problem,a concept formulated by Pease, Lamport, and others [10,11]. It was first proposed by Castro et al. [12] in 1999. Its core working principle is to achieve efficient consensus through multi-stage message interaction and division of node roles in an asynchronous network. When there are n nodes in the network system, it allows the existence of f Byzantine nodes, and the system can still operate normally, provided that the condition $n \geq 3f + 1$ is satisfied. However, traditional PBFT suffers from a communication complexity of $O(n^2)$. As the number of nodes in the system grows, communication volume increases drastically, leading to a decline in system efficiency. Therefore, the applicability of traditional PBFT is relatively low for the distributed network of unmanned aerial vehicle (UAV) swarms with limited computing resources.

In recent years, research endeavors on PBFT have predominantly concentrated on aspects such as scalability optimization, security enhancement and performance improvement. Numerous academics have put forth diverse approaches to enhance PBFT's performance, enabling its deployment in large-scale dynamic networks or networks with low computing power, thus expanding its application scenarios. Li et al. [13] introduced a novel "group-based" PBFT algorithm, which categorizes all nodes into distinct groups. Each group elects a primary node, reaches consensus internally first, and then achieves inter-group consensus. This method can markedly boost communication efficiency in scenarios with a substantial number of nodes, though it encounters certain issues regarding security performance.Additionally, Lao et al. [14] proposed a location-based scalable PBFT consensus algorithm. Given that mobile nodes generally have weaker computing capabilities than fixed nodes and are more prone to becoming malicious nodes, this algorithm can considerably reduce consensus overhead by selecting fixed and trusted nodes for consensus, albeit at the cost of increasing centralization. Wang et al. [15] devised a credit-based improved PBFT consensus algorithm (CPBFT). By modifying the original architecture and introducing a credit coefficient, nodes with high credit coefficients are chosen as primary nodes, which enhances the algorithm's throughput and security to some extent. Xu and Wang [16] and Yang et al. [17] utilized hash algorithms to differentiate consensus nodes in the PBFT consistency protocol, thereby avoiding excessive inter-node communication and reducing communication complexity. However, this approach fails to identify Byzantine nodes.

Currently, there exists a wide variety of applied research on the use of blockchain for identity authentication in distributed networks. Cui et al. [18] constructed a local-public hybrid blockchain architecture in the IoT, classifying nodes in the IoT into cluster nodes and ordinary nodes. The local chain undertakes the identity authentication of ordinary nodes, while the public chain is tasked with the identity authentication of cluster nodes. NVSVS [19], Roman et al. [20] and Qashlan et al. [21] have deployed blockchain on servers with higher computing power to achieve efficient identity authentication. Haque et al. [22] enhanced network scalability by integrating the Delegated Proof of Stake (DPoS) consensus algorithm in blockchain with sharding technology. Kong et al. [23] proposed a stateless blockchain based on triple aggregated sub-vectors and a dynamic Proof of Authorized Trust (PoAT) consensus mechanism. By periodically selecting authorized nodes randomly, this mechanism ensures the credibility of mutual authentication among UAV nodes. Zhang [24] divided UAV identity authentication into two phases: a main blockchain for identity authentication of UAVs prior to network access, and a sub-blockchain for identity authentication among UAVs during mission execution, thus tackling various security threats.

In summary, due to the high-frequency data interaction required by UAV swarms in missions such as military strikes and reconnaissance, the use of blockchain for traditional distributed identity authentication faces the problem of an imbalance between security and efficiency. In particular, the PBFT algorithm in blockchain has drawbacks including high communication complexity and difficulty in eliminating malicious nodes, making it unable to adapt to the resource-constrained and dynamic topology characteristics of UAVs. Therefore, this paper designs a lightweight CG-PBFT algorithm and conducts multi-dimensional comparisons with other improved PBFT consensus algorithms, as shown in Table 1. These comparisons demonstrate its high adaptability to UAV swarm networks, enabling secure, efficient, and scalable identity authentication for UAV swarms.

Table 1: Algorithm comparison

Alg type	**Rep mech**	**Com cplx**	**Group meth**	**Sig tech**	**Mal node rem**	**Dyn adapt**	**UAV scen fit**
PBFT	None	$O(n^2)$	None	SNS	Pass. Tol	Poor	Poor
HotStuff	None	$O(n)$	None	SNS+CAC	Pass. Tol	Inferior	Inferior
mPBFT	None	$O(n^2)$	None	SNS	Pass. Tol	Inferior	Inferior
CPBFT	S-Dim. Rep	$O(n^2)$	None	SNS	Scr. Avd	Moderate	Moderate
RBFT	M-Dim. Rep	$O(n^2)$	None	SNS	Scr. Avd	Moderate	Moderate
G-PBFT	None	$k \cdot O(m^2)$	Sta.-Pos.Grp	SNS	Scr. Avd	Poor	Poor
CG-PBFT	4L-RD+RL	$O(n)$	Rep.-Dyn. Grp	2S-BLSAS	DS+AE	Excellent	Excellent

Note: *Abbreviations:* Alg Type = Algorithm Type; Rep Mech = Reputation Mechanism; Com Cplx = Communication Complexity; Group Meth = Grouping Method; Sig Tech = Signature Technology; Mal Node Rem = Malicious Node Removal; Dyn Adapt = Dynamic Adaptability; UAV Scen Fit = UAV Scenario Fitness; Pass. Tol = Passive Tolerance; Scr. Avd = Screening Avoidance; 4L-RD+RL = 4-Level Reputation Dynamic Grading + Role Linkage; SNS = Single-Node Signature; CAC = Chain Aggregation Confirmation; 2S-BLSAS = Two-Stage BLS Aggregate Signature; DS+AE = Dynamic Screening + Active Expulsion; Sta.-Pos.Grp = Static Position Grouping; Rep.-Dyn.Grp = Reputation-Driven Dynamic Grouping; S-Dim. Rep =Single-Dimensional Reputation; M-Dim. Rep = Multi-Dimensional Reputation.

As shown in Table 1, the CG-PBFT consensus algorithm proposed in this paper has the following three unique contributions compared with other PBFT consensus algorithms:

1. Different from the single credit coefficient of CPBFT and the fixed grouping of G-PBFT, the CG-PBFT consensus algorithm designs a collaborative mechanism of four-level reputation grading and dynamic grouping targeting the "dynamic topology + resource-constrained" characteristics of UAV swarms. This mechanism improves system adaptability and consensus efficiency in dynamic scenarios, effectively reduces communication overhead, and enhances the stability of node collaboration.

2. Distinguished from existing algorithms that only use aggregate signatures in a single phase, the CG-PBFT consensus algorithm deeply integrates BLS aggregate signatures into the two-phase consensus of "intra-group + inter-group". Meanwhile, it optimizes the signature verification process to improve consensus efficiency.

3. A reputation-role linked adjustment strategy is proposed, which designs mechanisms for decreasing rewards for high-reputation nodes, promotion of ordinary nodes, and expulsion of malicious nodes. This strategy avoids the monopoly of oligarch nodes, optimizes the network architecture, and improves network security.

2 Preliminary Knowledge

2.1 PBFT Consensus Algorithm

The Practical Byzantine Fault Tolerance (PBFT) algorithm serves as a versatile solution in distributed networks, capable of tolerating malicious nodes and achieving consensus. Generally, in a distributed network with n nodes, when the condition $n \geq 3f + 1$ is satisfied, the network can still reach a consensus even with f malicious nodes, thus guaranteeing data confidentiality and credibility. The consensus process of the PBFT algorithm is segmented into the following five stages, as shown in Fig. 1:

(1) Request phase: The client transmits a request message to the primary node, which contains the request content, timestamp, and the client's identity information.

(2) Pre-prepare phase: Upon receiving and verifying the client's request message, the primary node assigns a corresponding proposal number and then dispatches a pre-prepare message to all replica nodes in the network.

(3) Prepare phase: After receiving and successfully verifying the pre-prepare message, the replica node sends prepare messages to the primary node and other replica nodes. Simultaneously, it validates the prepare messages from other replica nodes. When the node receives $2f + 1$ verified messages, it proceeds to the Commit phase.

(4) Commit phase: The primary node and replica nodes generate confirmation messages and broadcast them to all other nodes. After receiving these messages, the nodes perform verification. Once at least $2f + 1$ messages pass the verification, they enter the Reply phase.

(5) Reply phase: The primary node and replica nodes send reply messages to the client. When $f + 1$ nodes have identical responses, it signifies that the client's request has succeeded and consensus has been achieved.

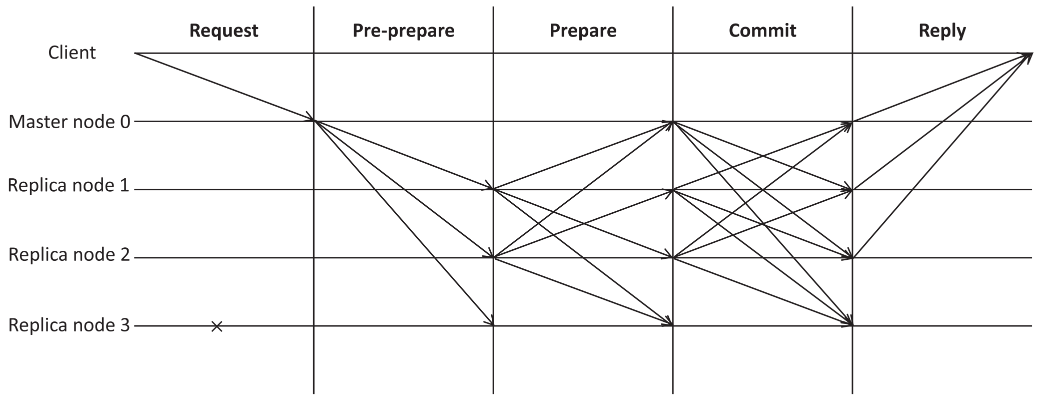

Figure 1: PBFT consensus process

During the entire consensus process, the messages received by nodes at each stage are stored in the local log, and Byzantine nodes do not feed back any information to other normal nodes. As shown in Fig. 1, replica node 3 is a Byzantine node. However, due to the PBFT consensus mechanism, as long as the number of Byzantine nodes does not exceed f under the condition of $n \geq 3f + 1$, the distributed network can still complete the consensus.

Generally, the entire consensus process requires nodes to finish within a specific time period. However, if the primary node turns malicious or a consensus timeout occurs due to poor network conditions, the view change protocol will be triggered to replace the primary node and update the view number, thereby ensuring the liveness and security of the entire distributed network.

2.2 BLS Aggregate Signature

The BLS aggregate signature is a digital signature scheme based on elliptic curve pairing proposed by Boneh et al. [25] in 2001. It can realize the functions of signature generation and verification by using bilinear pairing, and its signature has the characteristics of short signature, security and uniqueness. The basic process of signature is as follows:

(1) Key Generation: Select a random number as the private key SK, and compute the public key PK as $PK = SK \times G$, where G is the generator point of the elliptic curve.

(2) Signature Generation: For the message m to be signed, firstly use a hash function H to map it to a point on the elliptic curve, denoted as $q = H(m)$. Then, multiply the resulting point by the private key SK to obtain the signature $S = SK \times q$.

(3) Signature Verification: Calculate the point $q = H(m)$ mapped to the curve in the same way as in key generation and signature generation. Then, verify the signature's validity by checking whether the pairing equation $e(PK, q) = e(G, S)$ holds, where e is the elliptic curve pairing function. If the equation holds, the signature is valid, indicating the message m is indeed signed by the signer who owns the corresponding private key SK; otherwise, the signature is invalid.

Introducing BLS aggregate signatures into the blockchain network of unmanned aerial vehicle (UAV) swarms can significantly reduce the transmission volume of signature data during communication among UAV swarms, lower communication latency, and improve consensus efficiency.

3 Design of Light-Weight Blockchain-Based Distributed Networking Identity Authentication System for UAV Swarms

3.1 Design of Lightweight Blockchain-Based Network Architecture for UAV Swarms

This paper constructs a network architecture, as shown in Fig. 2, which consists of three core modules: the ground base station, the UAV swarm, and the blockchain. The ground base station module is responsible for initial network configuration, task issuance, generating public-private key pairs and data certificates for UAVs in the network, and sending public keys and data certificates of the UAVs to the blockchain module for storage. The UAV swarm module is composed of UAVs that have been registered through the ground base station, and its main function is to receive the private keys and task requests sent from the ground base station. The blockchain module is composed of two sub-units, namely the ground base station chain unit and the UAV swarm chain unit. The two sub-units share data. The ground base station chain unit is used for mutual communication and distributed identity authentication between the UAV swarm module and the ground base station, while the UAV swarm chain unit is used for mutual communication and distributed identity authentication within the UAV swarm module.

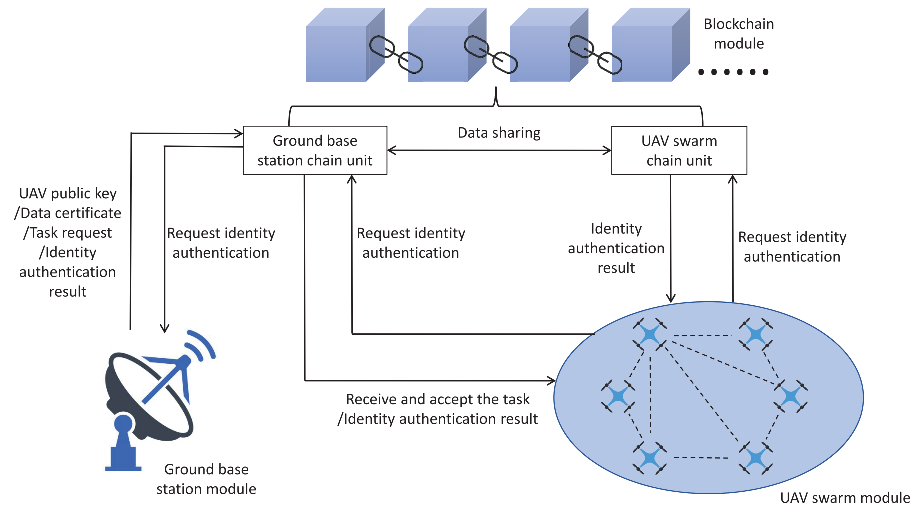
Blockchain module
......
Ground base station chain unit
Data sharing
UAV swarm chain unit
UAV public key /Data certificate /Task request /Identity authentication result
Request identity authentication
Request identity authentication
Identity authentication result
Request identity authentication
Receive and accept the task /Identity authentication result
Ground base station module
UAV swarm module

Figure 2: Drone cluster blockchain network architecture

3.2 Design of UAV Identity Authentication Process

Step 1: UAV Network Access Registration

Before a UAV accesses the network, it must first register with the ground base station module to obtain an asymmetric public-private key pair $\{SK_i, PK_i\}$ and a corresponding digital certificate $DigiCert_i$. The digital certificate $DigiCert_i$ contains the identity information ID_i of the UAV node, the public key PK_i, the validity period VP_i, the initial reputation score $C_{initial}$, and the signature $Sign_i$. Among them, the UAV's private key SK_i is kept by itself, and the digital certificate $DigiCert_i$ of each UAV is written into the blockchain module for storage by the ground base station module through the ground base station chain unit, which is used for subsequent identity authentication. After the UAV registration is completed, the ground control station (GCS) and the UAV (UAV_i), as well as between UAV (UAV_i) and UAV (UAV_j), use hash functions to calculate mutually verifiable shared security keys ($Key_{Gi} = H(SK_i \cdot PK_G)$; $Key_{ij} = H(SK_i \cdot PK_j)$) respectively, which are used to encrypt and decrypt communication content, ensuring the confidentiality and integrity of communication.

Step 2: Drone Identity Authentication

When the UAV cluster modules conduct internal communication, identity authentication must be performed first. At this point, the sender needs to package its own digital certificate ($DigiCert_i$), message (M) and timestamp (T), encrypt them with its own security key in the form of $\{Key_{ij} \cdot PK_i \cdot H(DigiCerti \| M \| T \|)\}$, and send them through the UAV cluster chain unit. After receiving the information, other UAVs will verify the message using the security key for interactive verification to ensure the credibility of the sender.

Step 3: The Addition Of New Drone Nodes

When a new drone intends to join the network, it first registers with the ground base station module, which generates a digital certificate ($DigiCert_i$) and a corresponding pair of public and private keys $\{SK_i, PK_i\}$. Then, the ground base station module sends these to the current central drone. After that, the new drone initiates a network access request to the current central drone. Once the central drone verifies that the digital certificate ($DigiCert_i$) is correct, it broadcasts a message to the entire drone cluster module and the ground base station module. The ground base station module then writes the updated information into the blockchain module for storage.

Step 4: Deletion Of UAV Nodes

In the UAV cluster module, when a UAV node needs to be deleted or withdrawn from the network, the ground base station module adds an invalidation label to the node's digital certificate ($DigiCert_i$) and uploads it to the blockchain module. Then, it broadcasts the invalidation message of the digital certificate ($DigiCert_i$) to the UAV cluster module. When the system encounters an access request with this $DigiCert_i$ again, it will reject the service.

In this system, by leveraging the immutability of blockchain to record the public keys, digital certificates, and behaviors of drones, any attempt to tamper with historical records on the chain can be detected at any time. Meanwhile, the identity authentication mechanism and message encryption method ensure the security of drone identities as well as the confidentiality and integrity of messages. Therefore, the security and credibility of the system have been significantly enhanced.

4 Design of CG-PBFT Consensus Algorithm Model

4.1 Node Reputation Grading And Grouping Mechanism

4.1.1 Node Reputation Grading

In the traditional PBFT consensus algorithm, the selection of the primary node is determined by the current view number v. When there are n nodes, the calculation formula is $p = v \mod n$. Malicious nodes may also be selected as primary nodes, which makes it difficult to meet the security requirements of the UAV cluster network. Therefore, a reputation score C is introduced to evaluate nodes and optimize the selection of primary nodes. The reputation score C is divided into four intervals, namely high-reputation nodes, ordinary-reputation nodes, low-reputation nodes, and suspicious-reputation nodes, as shown in Table 2. When each UAV registers and accesses the network, it is assigned an initial reputation score $C_{initial}$. The reputation score is increased or decreased based on the node's behaviors during the consensus process. When a node's reputation score is lower than C_{low}, it is identified as a suspicious node and kicked out of the network. This ensures that all nodes in the consensus process are trustworthy, ultimately guaranteeing the security of the system.

Table 2: Node reputation score grading

Reputation grading	**Range of reputation scores**
High-reputation score nodes	$[C_{high}, +\infty]$
Ordinary-reputation score nodes	$[C_{initial}, C_{high}]$
Low-reputation score nodes	$[C_{low}, C_{initial}]$
Suspicious-reputation score nodes	$[0, C_{low}]$

Among them, both C_{low} and C_{high} are thresholds, which can be dynamically adjusted according to the increase or decrease of reputation scores.

4.1.2 Node Grouping Mechanism

According to the four-level grouping criteria based on reputation scores, UAV cluster nodes are divided into four grades: high-reputation UAV nodes, ordinary-reputation UAV nodes, low-reputation UAV nodes, and suspicious UAV nodes. Based on this four-level classification, the specific grouping mechanism is shown in Fig. 3. All high-reputation UAV nodes form a leading group, among which the UAV node with the highest reputation score serves as the primary node and is responsible for leading the entire leading group. The remaining high-reputation UAV nodes act as leading nodes, each commanding a led group. The members of the led groups, as slave nodes, consist of randomly assigned ordinary-reputation UAV nodes and low-reputation UAV nodes. Suspicious UAV nodes with reputation scores lower than C_{low} are excluded from the network and do not participate in the consensus process.

Meanwhile, in accordance with the requirements of the consensus mechanism, the number of nodes in the leading group must satisfy $n \geq 4$. When there are insufficient high-reputation score nodes, the nodes with the highest reputation scores from the ordinary-reputation score nodes will be selected to supplement the leading group. Additionally, the maximum number of nodes in each led group, $n_{max} (n_{max} \geq 3)$, is dynamically set, and the acceptance of new nodes into the group will be stopped once this maximum number is reached.

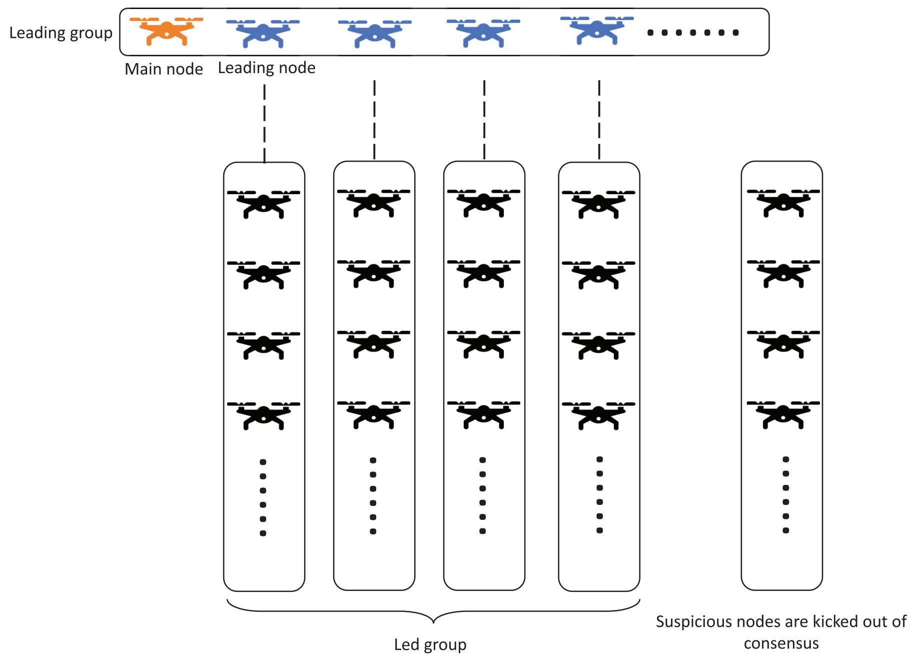

Figure 3: A four level grouping architecture based on reputation points of drone clusters

4.2 Design of Reputation Score Evaluation Model

In the consensus process, considering the characteristics of UAV cluster networks, such as frequent dynamic changes, unstable message routing, and the vulnerability of UAVs to electromagnetic interference and malicious attacks, even when most nodes actively and normally participate in the consensus, there may be issues where some nodes fail to participate in the consensus in a timely or complete manner due to network problems, as well as the existence of malicious nodes. In response to the above situations, a reputation score evaluation model for nodes is designed.

Reputation score evaluation model:

$$C_{i,k} = \frac{\sum_{1}^{k-1} C_{i,j}}{k-1} + \alpha * R_i + \beta * A_i + \gamma * B_U \tag{1}$$

Among them, $C_{i,k}$ denotes the reputation score of the i-th consensus node in the k-th round of consensus; similarly, $C_{i,j}$ represents the reputation score of the i-th consensus node in the j-th round of consensus; R_i denotes the direct trust value of the consensus node; A_i is the incentive and punishment parameter for the node's participation in consensus behaviors; B_U is the indicator of the role played by the node in the consensus; α, β, and γ are the weight coefficients of each parameter, respectively. After the completion of each round of consensus, the node's reputation score is dynamically adjusted according to the evaluation model, thereby ensuring more reliable consensus completion.

The direct trust value of a node is expressed as:

$$R_i = \left(\frac{G - X}{Y + \mu}\right) * T_c \tag{2}$$

Among them, G is the number of times the node has successfully completed consensus, X represents the number of times the node has failed in the consensus, Y represents the total number of consensus times, and μ represents a very small number to prevent the denominator from being zero. T_c is the time decay factor of the node's historical reputation, which is expressed as:

$$T_c = e^{-\frac{t}{2}} \ (1 \le t \le n) \tag{3}$$

The direct trust value is a key indicator for evaluating the reliability of a node. The higher the direct trust value, the more reliable the node.

The incentive and punishment parameter of a node is expressed as:

$$A_i = \begin{cases} f(C_{i,k-1}) & Q = 1 \\ f(C_{i,k-1}) - \frac{2}{\tau} & Q = 0 \quad C_{i,k-1} \le C_{initial} \\ f(C_{i,k-1}) - \frac{1}{\tau} & Q = 0 \quad C_{initial} \le C_{i,k-1} \le C_{high} \end{cases} \tag{4}$$

Among them, Q is a quantitative representation of whether the node C_i has successfully completed the consensus, in the form of:

$$Q = \begin{cases} 1 & ConsensusSuccess \\ 0 & tConsensusFailure \end{cases} \tag{5}$$

$f(x)$ is expressed as:

$$f(x) = \frac{e^{\frac{-(x-C_{initial})}{\tau}}}{\tau\left(1 + e^{\frac{-(x-C_{initial})^2}{\tau}}\right)} \tag{6}$$

Among them, τ is the shape parameter, and $1/\tau$ is the peak value of the function.

When nodes participate in the consensus, the above formula is used for incentive and punishment control to better ensure the honest behavior of nodes. For high-reputation nodes serving as primary nodes or leading nodes, their rewards are reduced to prevent the emergence of oligopoly and avoid network centralization. For nodes with reputation scores around $C_{initial}$, their rewards are increased to encourage them to participate in the consensus more actively and honestly. When a node misbehaves, if it is a primary node or a leading node, its reputation score is directly reset to the initial score $C_{initial}$; for nodes with reputation scores lower than $C_{initial}$, the punishment intensity is increased to enhance the security of the overall network.

The role indicator of a node is expressed as:

$$B_U = \begin{cases} (C_{initial} - C_{high}) * \left(1 - e^{-\frac{k}{4}}\right) \\ (C_{initial} - C_{high}) * \left(1 - e^{-\frac{k}{6}}\right) \\ 0 \end{cases} \tag{7}$$

Due to the different tenure statuses of nodes in the consensus, the formula $(C_{initial} - C_{high}) \cdot (1 - e^{-k/4})$ is used when a node acts as the master node; the formula $(C_{initial} - C_{high}) \cdot (1 - e^{-k/6})$ is adopted when the node serves as the leading node; and $B_U = 0$ when the node only participates in the consensus, where k represents the number of rounds that the node has acted as the master node or leading node. The role indicator B_U is designed to prevent a single node from acting as the master node or leading node for a long time, avoid excessive concentration of power, and ensure the decentralized nature of the system. Meanwhile, it allows ordinary nodes to have the opportunity to improve their reputation scores through good consensus performance, thereby enabling them to act as master nodes or leading nodes. This guarantees the liveness of the system and contributes to its healthy operation.

4.2.1 Basis for Weight Setting

To address the repeatability issue of adjustable weights in the reputation score evaluation model, this paper clarifies the setting logic of each weight (α, β, γ, τ) based on the parameter design of reputation models in references [26,27] and combined with the core characteristics of UAV swarm networks, as detailed below:

1. α (Weight of direct trust value R_i): Its core function is to prioritize the long-term reliability of nodes. UAVs may encounter temporary consensus failures due to electromagnetic interference, while a hallmark of malicious nodes lies in persistent consensus failures. Consequently, α is configured to a relatively high value. This not only precludes the infiltration of malicious nodes when α is excessively low but also mitigates over-reliance on historical data when α is overly high, which would impede new nodes from enhancing their reputation through benign behaviors, thus ensuring that nodes with stable historical performance are prioritized as consensus nodes.

2. β (Weight of incentive-penalty parameter A_i): It is used to balance the penalty for malicious behaviors and the fault tolerance for non-malicious failures. Thus, β is set to a medium value. This both prevents non-malicious nodes from being excessively penalized for temporary failures (leading to an increased exit rate) when β is too high and avoids insufficient suppression of malicious nodes when β is too low.

3. γ (Weight of role indicator B_U): Its core goal is to avoid power concentration while maintaining consensus efficiency. Therefore, γ is set to a relatively low value to prevent high-reliability nodes from gaining excessively high reputation due to assuming key roles when γ is too high, it would hinder role rotation and result in centralization.

4. τ (Shape parameter of incentive function $f(x)$): It is used to optimize the incentive intensity for reputable nodes. It not only prioritizes encouraging initial reputable nodes to actively participate in consensus but also avoids excessively extreme incentives or penalties for nodes with extremely high or low reputation.

In this paper, $\alpha = 0.5$, $\beta = 0.3$,$\gamma = 0.2$, and $\tau = 5$ are set to suit UAV swarms with 30–100 nodes.

4.2.2 Improvement of Consensus Process

After the grouping of UAV nodes is completed, all UAV nodes in the network are divided into K groups, and BLS aggregate signatures are introduced into the consensus process to further reduce communication overhead. The entire consensus process is mainly divided into two parts: intra-group consensus and inter-group consensus. In the intra-group consensus, when the consensus starts, the master node in the entire network delivers messages to each leading node, and each leading node then forwards the messages to the slave nodes within their respective groups. Consensus is conducted within each group, and finally, the messages are updated to each leading node. After the completion of intra-group consensus, the inter-group consensus begins. The master node in the leading group leads the leading nodes within the group to complete the consensus, and each leading node updates the messages to the master node, thus finishing the entire consensus process, as shown in Fig. 4.

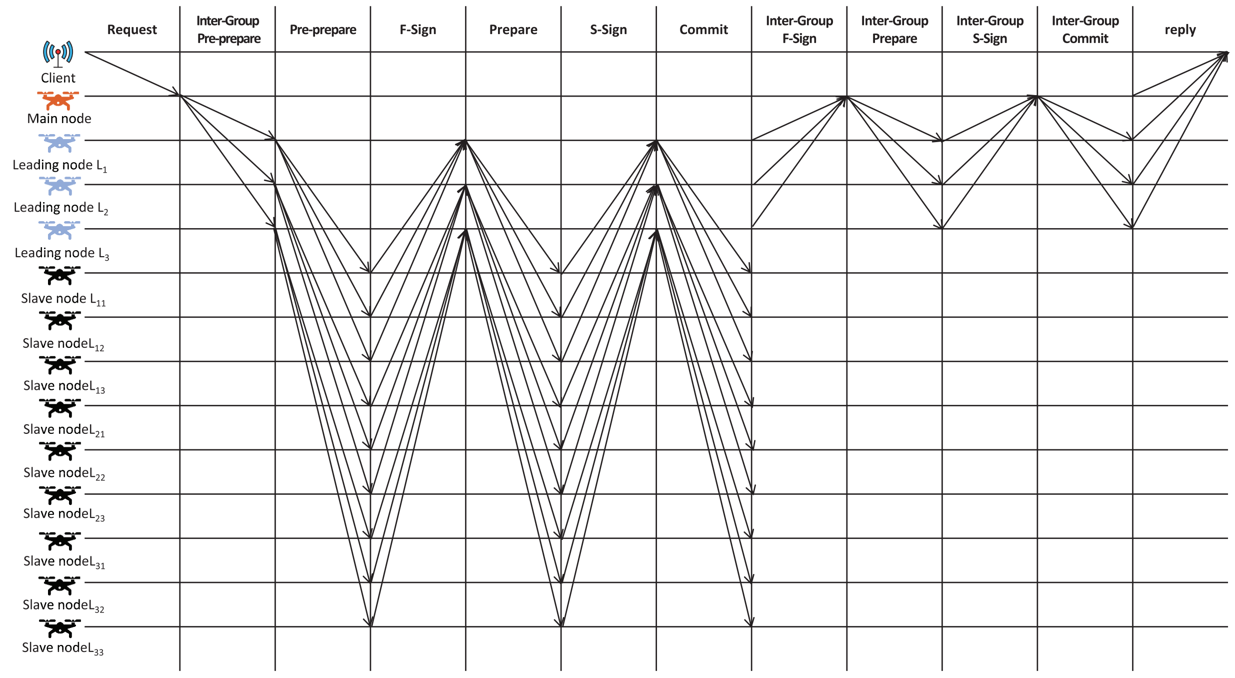

Figure 4: Consensus process of drone cluster nodes

(1) Request phase. The client sends a request to the master node, with the message formatted as $\langle Request, O, T, ID_c, Sign_c\rangle$. Here, O denotes the operation requested by the client, T is the current timestamp, ID_c represents the client's identity information, and $Sign_c$ is the client's signature.

(2) Inter-Group Pre-prepare phase. After the master node receives and correctly verifies the request message, it sends a pre-prepare message to the leading nodes of each group. The message format is $\langle \textit{M-Pre-prepare}, v, T, M_z, ID_Z, Sign_m\rangle$, where M-Pre-prepare is the message identifier of the master node, v is the view number, T is the timestamp, M_z is the request message of the master node, ID_Z is the identity information of the master node, and $Sign_m$ is the signature of the master node.

(a) Pre-prepare phase. The leading node sends a pre-prepare message to the slave nodes in its group, using the format$\langle \textit{L-Pre-prepare}, v, T, M_{L_i}, ID_{L_i}, Sign_{L_i}\rangle$. Here L-Pre-prepare is the message identifier of the leading node, M_{L_i} is the request message from leading node Li, ID_{L_i} is the leading node's identity information, and $Sign_{L_i}$ is the leading node's signature.

(b) F-Sign phase. Upon receiving the message sent by the leading node, the slave nodes in the group verify it. If the verification is successful, they send a message to the leading node with the format $\langle \textit{C-F-Sign}, v, T, M_c, ID_i, Sign_i\rangle$, where ID_i is the identity information of the slave node, C-F-Sign is the slave node's message identifier, M_c is the verification result message of the slave node, and $Sign_i$ is the signature of the slave node. When the leading node receives messages from $2f_m + 1$ slave nodes, it verifies them (where f_m is the maximum number of Byzantine nodes that can be accommodated in the leading group). If the verification is successful, it aggregates the received message signatures based on the BLS signature to generate an aggregated signature $Sign_{p_a}$.

(c) Prepare phase. The leading node sends a message with an aggregated signature to the slave nodes within the group, and the message format is $\langle \textit{L-Prepare}, v, T, M_{L_i}, ID_{L_i}, Sign_{pa}\rangle$. The slave nodes verify the correctness of the message and the validity of the aggregated signature, and also verify whether the number of nodes that generated the aggregated signature is $2f_m + 1$. After the verification is passed, they proceed to the next phase.

(d) S-Sign phase. After the slave nodes in the group successfully verify the signature, they send an S-Sign message to the leading node, with the message format being $\langle \textit{C-S-Sign}, v, T, M_c, ID_i, Sign_i \rangle$. When the leading node receives messages from $2f_m + 1$ slave nodes, it performs verification. If the verification is successful, the leading node also aggregates the message signatures based on the BLS signature to generate an aggregated signature $Sign_{p_b}$.

(e) Commit phase. The leading node broadcasts a message with an aggregated signature to all slave nodes within the group, and the message format is $\langle \textit{L-Commit}, v, T, M_{L_i}, ID_{L_i}, Sign_{pb} \rangle$. After receiving the message, the slave nodes verify the correctness of the message and the validity of the aggregated signature, and also check whether the number of nodes that generated the aggregated signature is $2f_m + 1$. Upon successful verification, the intra-group consensus is completed, and the process proceeds to the inter-group consensus.

(3) Inter-Group F-Sign phase. After each group completes the intra-group consensus, the leading node sends a message to the master node in the leading group, with the message format being $\langle \textit{M-F-Sign}, v, T, M_{L_i}, ID_{L_i}, Sign_{L_i} \rangle$. When the master node receives messages from $2f_h + 1$ leading nodes, it verifies them (where f_h is the maximum number of Byzantine nodes that the leading group can accommodate). If the verification is successful, it aggregates the received message signatures based on the BLS signature to generate an aggregated signature $Sign_{pc}$, and then proceeds to the next phase.

(4) Inter-Group Prepare phase. The master node broadcasts broadcasts the message to all leader nodes in the group. The message format is $\langle \textit{M-Prepare}, v, T, M_z, ID_Z, Sign_{pc} \rangle$. Then, the leader nodes verify the correctness of the message and the validity of the aggregate signature, and check whether the number of nodes that generated the aggregate signature is $2f_h + 1$. After passing the verification, they proceed to the next phase.

(5) Inter-Group S-Sign phase. After the leading nodes in the group successfully verify the signature, they send an S-Sign message to the master node, with the message format being $\langle \textit{L-S-Sign}, v, T, M_{L_i}, ID_{L_i}, Sign_i \rangle$. When the master node receives messages from $2f_h + 1$ leading nodes, it performs verification. If the verification is successful, the master node aggregates the message signatures based on the BLS signature to generate an aggregated signature $Sign_{pd}$.

(6) Inter-Group Commit phase. The master node broadcasts a message with an aggregated signature to all leading nodes in the group, and the message format is $\langle \textit{M-Commit}, v, T, M_z, ID_Z, Sign_{pd} \rangle$. After receiving the message, the leading nodes verify the correctness of the message and the validity of the aggregated signature, and also check whether the number of nodes that generated the aggregated signature is $2f_h + 1$. Upon successful verification, they proceed to the next phase.

(7) Reply phase. The master node and leading nodes send messages to the client, with the formats being $\langle \textit{M-Reply}, v, T, z, M_z, ID_Z \rangle$ and $\langle \textit{L-Reply}, v, T, z, M_z, ID_{L_i} \rangle$ respectively. Here, ID_Z and ID_{L_i} are the identity information of the master node and the leading node, respectively, and z is the execution result of the client's request. After receiving the messages, the client verifies them. If at least $f_h + 1$ nodes respond identically, the verification is successful, indicating that the request initiated by the client has succeeded and the consensus is completed.

4.2.3 View Switch

Although the probability that the master node or leading node selected by the reputation score evaluation mechanism becomes a Byzantine node is greatly reduced, in order to ensure the liveness and security of the system, this paper still designs a view switch protocol for cases where the master node or leading node is down or the consensus node exhibits Byzantine behavior, and the flow chart is shown in Fig. 5.

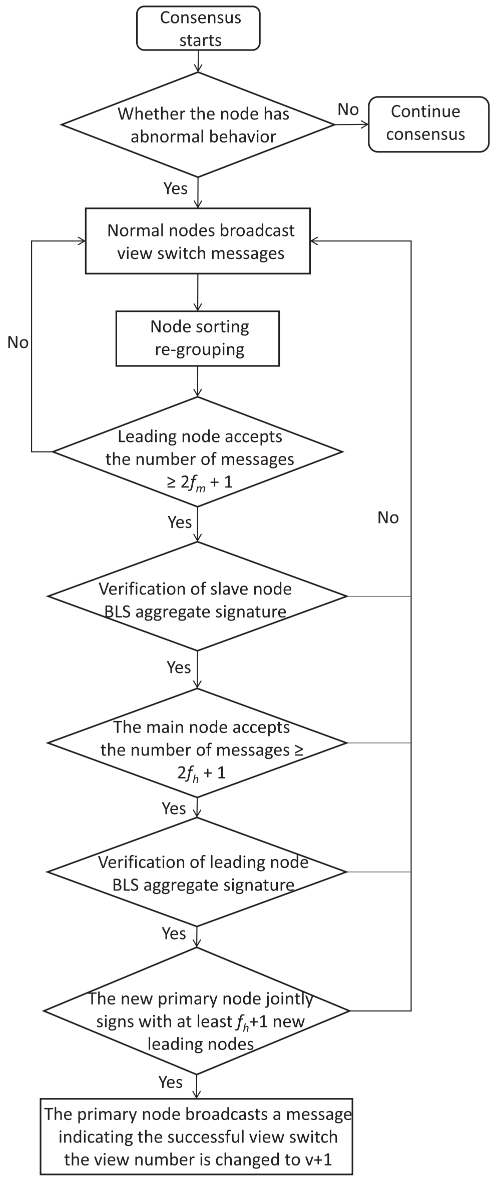
Consensus starts
Whether the node has abnormal behavior
No
Continue consensus
Yes
Normal nodes broadcast view switch messages
Node sorting re-grouping
No
Leading node accepts the number of messages ≥ $2f_m + 1$
Yes
No
Verification of slave node BLS aggregate signature
Yes
The main node accepts the number of messages ≥ $2f_h + 1$
Yes
Verification of leading node BLS aggregate signature
Yes
The new primary node jointly signs with at least f_h+1 new leading nodes
Yes
The primary node broadcasts a message indicating the successful view switch the view number is changed to v+1

Figure 5: View switching process

(1) When a node in the network experiences a timeout while waiting, the view switch protocol is triggered. At this point, new master nodes and leading nodes are selected in sequence based on the reputation score rankings obtained from the previous round of consensus, with those having higher reputation scores taking up the positions. After the selection is completed, the normally operating nodes in the network continue to broadcast information, with the message format being $\langle \textit{View-Change}, v, T, ID_Z, ID_{L_i}, Sign_i \rangle$. Here, v is the current view number, T is the current timestamp, ID_Z and ID_{L_i} are the identity information of the new master node and leading node respectively, and $Sign_i$ is the signature of the node issuing the message.

(2) After triggering the view change protocol, the view number is updated to $v + 1$. The view change is successful if and only if the following conditions are satisfied: the new group leader nodes receive messages from $2f_m + 1$ follower nodes within their respective groups; the primary node receives messages from $2f_h + 1$ group leader nodes; and the verification of the relevant aggregate signatures succeeds. Subsequently, the new primary node jointly signs with at least $f_h + 1$ new group leader nodes and disseminates the view change success message, which is formatted as $\langle \textit{View-Change-Success}, v, T, ID_Z, Sign_m, Sign_{List}, Sign_{Li} \rangle$ Herein, v denotes the new view number, T is the timestamp of the new view, $Sign_m$ and $Sign_{Li}$ represent the signatures of the new primary node and group leader nodes, respectively, and $Sign_{List}$ is the list of joint signatures. After receiving and successfully verifying the message, other nodes enter the new view.

4.3 *Proof of the Byzantine Fault Threshold of CG-PBFT under the Partially Synchronous Model*

4.3.1 Model Assumptions

Based on the characteristics of the partially synchronous network and the design logic of the CG-PBFT algorithm, the following core assumptions are specified to lay the foundation for the threshold proof:

Partially Synchronous Network Assumption: There exists an unknown but fixed upper bound Δ on message transmission delay, and the message delivery time between all honest nodes does not exceed Δ; the node clocks satisfy the weak synchronization constraint, i.e., the clock skew between any two honest nodes is no more than Δ (a global clock is not required, and only the consensus blockage caused by clock discrepancies needs to be avoided).

Node and Behavior Assumption: Let the total number of nodes in the system be N, where the number of Byzantine nodes is f and the number of honest nodes is $H = N - f$; additionally, the reputation score evaluation mechanism can exclude malicious nodes with a reputation score below C_{low} from the consensus participation set, ensuring that subsequent consensus proceeds only among valid nodes.

CG-PBFT Architecture Assumption: After node grouping, the number of group leader nodes is m ($m \geq 4$; if insufficient, high-reputation score nodes are selected from ordinary reputable nodes to supplement); there are k groups of follower node groups, with a fixed number of g nodes per group ($g \geq 3$, $N = m + g \times k$); the BLS aggregate signature satisfies the unforgeability and verifiability of elliptic curve cryptography (ECC), i.e., malicious nodes cannot forge the aggregate signature of honest nodes, and honest nodes can verify the signature validity through the pairing function $e(PK, q) = e(G, S)$.

4.3.2 Key Parameters and Consensus Validity Conditions

To quantify the fault tolerance capability of hierarchical consensus, the following parameters are defined as shown in Table 3:

Table 3: Definitions and constraints of key parameters

Parameter	Definition	Constraint relationship
f_h	HL BN Count (in GLSet)	$0 \leq f_h < m$
H_h	HL HN Count (in GLSet)	$H_h = m - f_h$
F_i	BN Count in i-th FG	$0 \leq F_i < g + 1$
H_i	HN Count in i-th FG	$H_i = g + 1 - F_i$
$Consensus_{Valid}$	Consensus validity condition	Requires $\geq 2F + 1$ HN signatures (F denotes UB of BNs in CL)

Note: *Abbreviations*: HL = High-level; GLSet = Group Leader Set; FG = Follower Group; BN = Byzantine Node; HN = Honest Node; UB = Upper Bound; CL = Corresponding Layer.

4.3.3 Derivation of the Byzantine Fault Threshold

The consensus process of CG-PBFT is divided into two levels: "intra-group consensus among follower groups" and "inter-group consensus across group leader nodes". The fault tolerance thresholds for the two levels of consensus are derived separately, and finally integrated into the overall system fault tolerance threshold.

(1) Intra-group Fault Tolerance Threshold of Follower Groups. Each follower group is required to reach an intra-group consistent result, generate an aggregate signature, and submit it to the group leader set. Under partial synchrony, it is necessary to ensure that even in the presence of malicious nodes, honest nodes can still reach a verifiable consistent result, i.e., $H_i \geq 2F_i + 1$. This condition guarantees that malicious nodes cannot forge a majority signature. Substituting $H_i = g + 1 - F_i$ into the inequality, the derivation proceeds as follows: $g + 1 - F_i \geq 2F_i + 1 \implies g \geq 3F_i \implies F_i \leq \frac{g}{3}$. In other words, a single follower group can tolerate at most $3/g$ Byzantine nodes.

(2) Inter-group Fault Tolerance Threshold of the Group Leader Set. The group leader set is responsible for aggregating results from all follower groups and generating a system-level consensus, whose fault tolerance capability directly determines the global system consistency. For the validity of inter-group consensus, the number of honest nodes in the group leader set must satisfy $H_h \geq 2f_h + 1$, which ensures that malicious group leader nodes cannot tamper with the inter-group consensus result. Substituting $H_h = m - f_h$ into the above condition, the derivation proceeds as follows: $m - f_h \geq 2f_h + 1 \implies m \geq 3f_h + 1 \implies f_h \leq \frac{m-1}{3}$. In other words, the group leader set can tolerate at most $(m-1)/3$ Byzantine nodes.

(3) Global Fault Tolerance Threshold of the System. The total number of Byzantine nodes in the system is $f = f_h + \sum_{i=1}^{k} F_i$. Combined with the system's group structure $N = m + g \times k$, the two-level fault tolerance constraints are substituted as follows: From the intra-group threshold derivation conclusion $g \geq 3F_i$, summing over all k follower groups yields $g \times k \geq 3\sum_{i=1}^{k} F_i$; and From the inter-group threshold derivation conclusion $m \geq 3f_h + 1$. Adding the two inequalities gives $m + g \times k \geq 3f_h + 1 + 3\sum_{i=1}^{k} F_i$. Substituting the total number of system nodes $N = m + g \times k$ and the total number of Byzantine nodes $f = f_h + \sum_{i=1}^{k} F_i$, the expression is rearranged to $N \geq 3f + 1 \implies f \leq \frac{N-1}{3}$. Thus, the final global Byzantine fault tolerance threshold of the system is $f \leq \frac{N-1}{3}$.

Furthermore, the system incorporates a reputation evaluation mechanism to dynamically exclude malicious nodes. Therefore, in practical consensus processes, the fault tolerance capability of the set of honest nodes will be superior to the theoretical threshold of the original system. This advantage stems from the fact that the reputation mechanism reduces the effective number of Byzantine nodes in the system, thereby enhancing the robustness of the consensus algorithm.

4.3.4 Proof of Safety and Liveness

(1) Consistency Proof: If the CG-PBFT algorithm satisfies the global fault tolerance condition $N \geq 3f + 1$, the intra-group fault tolerance condition of the group leader set $m \geq 3f_h + 1$, and the intra-group fault tolerance condition of the follower groups $g \geq 3F_i$, while the BLS aggregate signature satisfies validity and unforgeability, then CG-PBFT achieves consistency—meaning all honest nodes will eventually reach a consensus on the same proposal.

(a) Proof of Intra-group Consistency

Assume there exist two honest nodes P and Q in a certain follower group that confirm different intra-group proposals p and q, respectively. According to the intra-group consensus process, p must obtain BLS aggregate signatures from no fewer than $2F_i + 1$ honest nodes, and q must also obtain aggregate signatures from no fewer than $2F_i + 1$ honest nodes. Since the number of honest nodes in the group $H_i \geq 2F_i + 1$, there must be at least one honest node required to sign both proposals p and q, leading to an inevitable intersection between the two aggregate signatures. Furthermore, due to the unforgeability of BLS signatures, this honest node will not sign two conflicting proposals simultaneously. Therefore, the aggregate signatures of the two proposals cannot both pass verification, leading to a contradiction. Thus, intra-group consistency holds: all honest nodes in the same follower group will eventually confirm the same proposal result.

(b) Proof of Inter-group Consistency

Assume there exist two honest leader nodes P and Q in the group leader set that confirm different global aggregated proposals p and q, respectively. Based on intra-group consistency and the inter-group consensus process, the underlying data of the two aggregated proposals p and q is identical, and both must obtain aggregate signatures from no fewer than $2f_h + 1$ honest leader nodes in the group leader set. Furthermore, since the number of honest leader nodes in the group leader set $H_h \geq 2f_h + 1$, there exists at least one honest leader node in the group leader set that verifies both p and q simultaneously, resulting in an intersection between the signature sets of p and q. Due to the fact that honest leader nodes strictly adhere to the protocol, only sign proposals that are untampered and have consistent underlying data, and will inevitably reject signing conflicting proposals, a contradiction arises. Thus, inter-group consistency holds: all leader nodes in the group leader set will eventually confirm the same global aggregated result.

(c) Global Consistency Integration

The unique intra-group result is confirmed through intra-group consistency, and the unique global aggregated result is verified via inter-group consistency. Subsequently, the confirmed global aggregated result is synchronized to all nodes in the network by the leader nodes, thereby achieving the integration of global consistency.

(2) Liveness Proof: When the network delay eventually stabilizes within Δ, the CG-PBFT algorithm will complete consensus within a finite number of rounds and will not fall into infinite blocking.

(a) Termination of the Consensus Process

Based on the partial synchronous model and the CG-PBFT consensus process, under the premise of satisfying fault tolerance constraints and no malicious interference, any proposal initiated by an honest node will undergo the processes of proposal initiation (Δ), intra-group consensus (5Δ), inter-group transmission (Δ), inter-group consensus (5Δ), and global proposal synchronization (Δ). Eventually, global consensus will be completed within a time of 13Δ, ensuring that the consensus latency is finite.

(b) Liveness of View Change

When malicious interference exists or consensus times out, the view change protocol is triggered. Assuming the timeout waiting time is T, when an honest node undergoes the processes of broadcasting view change messages (Δ), message verification (2Δ), new view synchronization (Δ), and consensus completion (13Δ), the entire view change is guaranteed to end within $T + 4\Delta$, and both view change and global consensus will be completed within $T + 17\Delta$.

(c) Evasion of Malicious Node Blocking

Based on the reputation score evaluation mechanism, malicious nodes attempting to block the consensus process will experience a sharp drop in their own reputation scores and will eventually be kicked out of the network. This ensures that malicious nodes cannot block the consensus process for a long time.

5 Experimental Results and Analysis

This paper implements a multi-node and groupable blockchain system based on Python, simulating the identity registration and consensus process of an unmanned aerial vehicle (UAV) cluster on the blockchain.

In the simulation experiment, the network adopts a distributed mesh topology; nodes are uniformly distributed in a 10 km × 10 km area; the IEEE 802.11p communication protocol is used; and the delay model follows a Gaussian distribution. In this paper, the algorithms analyzed and compared with the proposed algorithm include the traditional PBFT algorithm, HotStuff algorithm, CPBFT algorithm, RBFT algorithm, and mPBFT algorithm (Modified PBFT). During the experiment, by changing the total number of nodes, the number of groups, and the number of malicious nodes in the network, the communication overhead, throughput, consensus delay, and security of the algorithm are analyzed. The experimental configurations are as follows: the hardware conditions are CPU Amd Ryzen 5 4600 h, 16 GB of memory, Windows 10 64-bit operating system, and the software environment includes PyCharm and MATLAB.

5.1 Analysis of Communication Overhead

Communication overhead refers to the total network resources consumed by message transmission between nodes to reach a consensus. The communication overhead of the PBFT algorithm consensus is based on its three consensus phases, namely the Pre-Prepare phase, the Prepare phase, and the Commit phase. The number of communications in these stages is $(N-1)$, $(N-1)^2$, and $N(N-1)$ respectively. Therefore, the communication overhead required for the traditional PBFT to complete one round of consensus is:

$$N_{PBFT} = (N-1) + (N-1)^2 + N(N-1) = 2N(N-1) \tag{8}$$

The CG-PBFT proposed in this paper reduces the communication overhead of consensus by implementing group consensus through reputation-based grouping of nodes participating in consensus and adding a signature phase in the consensus process.

Therefore, the communication overhead of CG-PBFT consensus includes the intra-group and inter-group consensus processes, which are respectively the Pre-Prepare phase, F-Sign phase, Prepare phase, S-Sign phase, and Commit phase of intra-group consensus, as well as the Inter-group Pre-Prepare phase, Inter-group F-Sign phase, Inter-group Prepare phase, Inter-group S-Sign phase, and Inter-group Commit phase of inter-group consensus. Due to the grouping mechanism of CG-PBFT, there will be schemes with different numbers of groups for consensus with the same number of nodes. To facilitate the explanation of its communication overhead, it is assumed that the number of nodes in the leading group is m, the number of nodes in each led group is $(N-m)/(m-1)$, and there are a total of $m-1$ led groups. Therefore, the communication overhead of CG-PBFT is the sum of the number of inter-group communications and the number of all intra-group communications.

Based on the above assumptions, the consensus between a led group and its leading node forms a set of intra-group consensus, involving a total of $(N-m)/(m-1)+1$ nodes. This intra-group consensus is divided into 5 stages, with the number of communications in each stage being $(N-m)/(m-1)$. Therefore, the total number of intra-group consensus communications N_1 across all groups is:

$$N_1 = 5(m-1)\frac{N-m}{m-1} = 5(N-m) \tag{9}$$

The inter-group consensus is the consensus of the leading group, involving a total of m nodes. It is divided into 5 stages, with the number of communications in each stage being $m-1$. Therefore, the number of inter-group consensus communications N_2 is:

$$N_2 = 5(m-1) \tag{10}$$

In summary, the communication overhead of CG-PBFT is:

$$N_{CG-PBFT} = N_1 + N_2 = 5(N-1) \tag{11}$$

It can be seen that compared with the traditional PBFT, the communication complexity of CG-PBFT is reduced from $O(n)^2$ to $O(n)$, and its communication overhead is significantly reduced, resulting in better communication performance in large-scale UAV cluster applications.

5.2 Throughput Analysis

In blockchain networks, throughput refers to the number of transactions that the network can process and confirm within a unit of time, which is usually expressed in TPS (Transactions Per Second). Its calculation formula can be expressed as:

$$TPS = \frac{Transactions}{\Delta t} \tag{12}$$

To verify the improvement of CG-PBFT in throughput, this paper sets up two experimental schemes. The first is to compare the throughput of the algorithm under different numbers of nodes but the same number of groups, and the second is to analyze the throughput changes of CG-PBFT under the same number of nodes but different numbers of groups. Based on the first experiment, the number of nodes set in this paper is 30, 40, 50, 60, 70, 80, 90, and 100 respectively, and the number of groups is set to 5. Consensus is carried out for each case, the experiment is repeated 100 times, and the calculated mean ± standard deviation value is taken as the final result. The comparison diagram is shown in Fig. 6:

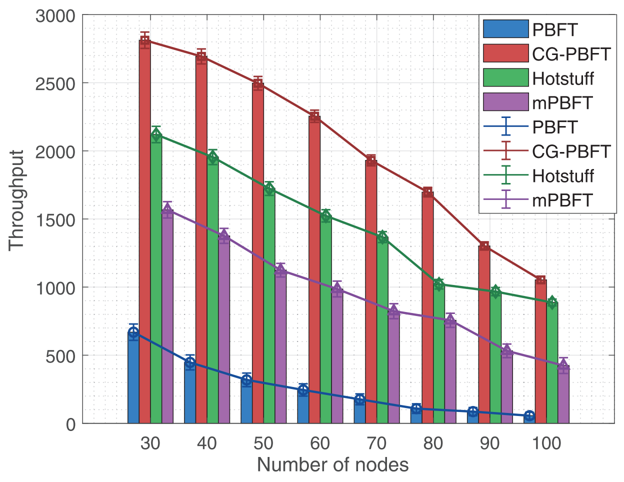

Figure 6: Experiment 1 throughput comparison

As shown in Fig. 6, as the number of nodes increases, the throughput of the four algorithms generally shows a downward trend, yet the throughput of CG-PBFT is notably higher than that of the other three algorithms under the same node count. This is because compared with the other three algorithms, CG-PBFT adopts a grouping consensus mechanism and uses aggregate signatures in the consensus phase, which greatly reduces its communication complexity, enabling the network system to maintain high throughput under a large number of nodes.

Based on Experiment 2, this paper selects 100 nodes and sets the number of groups to 4, 5, 10, 15, and 20 respectively. Consensus is performed 100 times for each setting, and the mean ± standard deviation value calculated is taken as the result. The results are presented in Fig. 7.

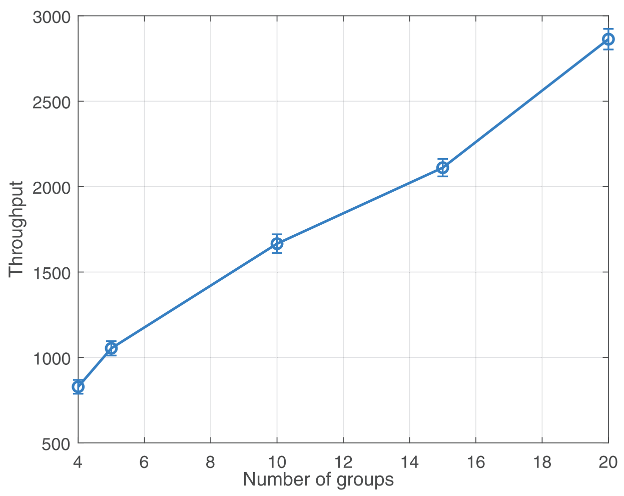

Figure 7: Experiment 2 throughput comparison

As can be observed from Fig. 7, when the number of nodes is identical, the throughput gradually rises as the number of groups increases. Therefore, for large-scale UAV clusters, according to the different throughput requirements of tasks, the maximum number of nodes in each group can be dynamically adjusted to achieve different numbers of groups, so as to adapt to the throughput required by tasks. CG-PBFT can be more suitable for the scenario application of large-scale UAV clusters.

5.3 Analysis of Consensus Delay

Consensus delay $Delay_i$ denotes the time interval from when a client initiates a request to when the request is completed in a blockchain network. Its formula is expressed as:

$$Delay_i = T_{finish} - T_{req} \tag{13}$$

In this paper, similar to the throughput analysis, two experiments are also conducted. The first experiment compares the consensus delay of each algorithm under varying node counts but a fixed group count. The node counts are set to 30, 40, 50, 60, 70, 80, 90, and 100, with the group count fixed at 5. Consensus is performed for each case, the experiment is repeated 100 times, and the calculated mean ± standard deviation value is taken as the final result. The second experiment investigates the changes in consensus delay of CG-PBFT under the same number of nodes but different numbers of groups. The node count is selected as 100, and the group counts are set to 4, 5, 10, 15, and 20. Consensus is carried out 100 times for each case, and the calculated mean ± standard deviation value is used as the result. The results of the two experiments are shown in Figs. 8 and 9.

Based on Experiment 1, it can be observed from Fig. 8 that while the consensus delay of all four algorithms rises with the increase in the number of consensus nodes, the consensus delay of CG-PBFT is considerably lower than that of the other three algorithms under the same node count and group count. From Fig. 9, it is evident that with the same number of consensus nodes, the consensus delay of CG-PBFT gradually decreases as the number of groups increases. According to the experimental results in

Figs. 8 and 9, it can be concluded that the consensus delay performance of CG-PBFT outperforms the other three algorithms. Moreover, with the same number of nodes, CG-PBFT can optimize the number of groups to reduce consensus delay, making it adaptable to diverse task scenarios.

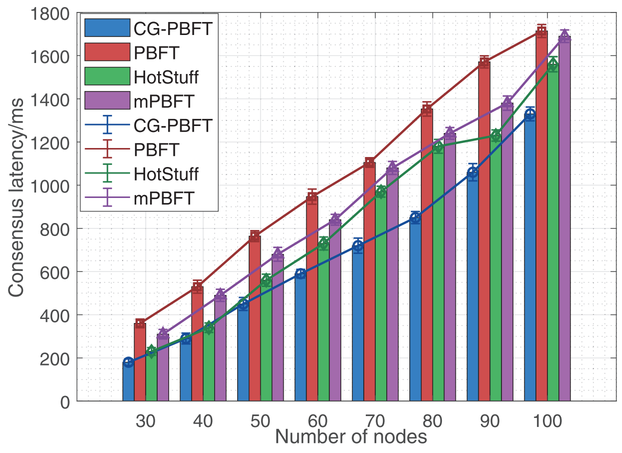

Figure 8: Experiment 1 comparison of consensus latency

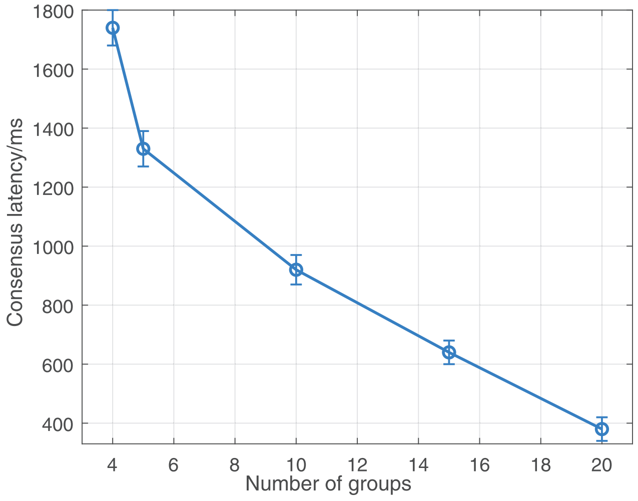

Figure 9: Experiment 2 comparison of consensus latency

5.4 Security Analysis

The CG-PBFT consensus algorithm designed in this paper incorporates a reputation score evaluation mechanism, selecting nodes with high reputation scores as primary nodes and leading nodes. Nodes with a reputation score below C_{low} will be excluded from the blockchain network. Compared with the traditional PBFT consensus algorithm, which allows malicious nodes to remain in the network system and thus reduce the security of the network system, the CG-PBFT consensus algorithm makes the blockchain network more secure, stable, and reliable. To further elaborate on the security of CG-PBFT, this paper sets up the following experiment: the number of nodes in the network is 30, 40, 50, and 60, among which different numbers of malicious nodes are set as 3, 7, 9, and 14 respectively (with different reputation scores but not lower than C_{low}). Malicious nodes are simulated by sending consensus messages with invalid signatures, delaying the transmission of critical messages, and other such methods. As the number of consensus rounds increases, the change in the number of malicious nodes in the network is observed, and the experimental results are shown in Fig. 10.

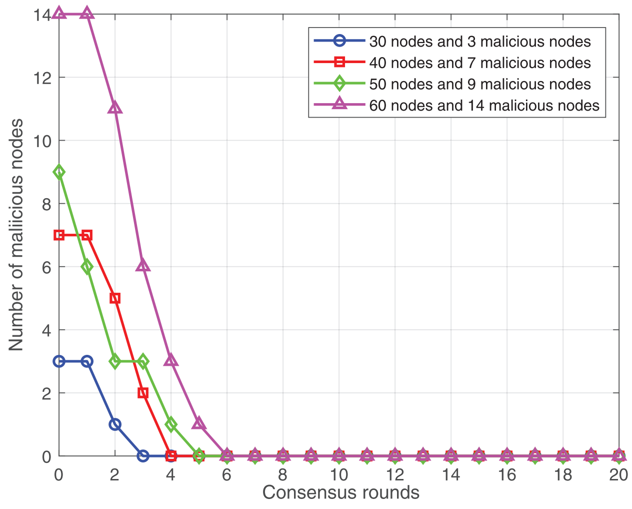

Figure 10: Comparison of the number of malicious nodes

According to the experimental results, the count of malicious nodes will gradually decline as the number of consensus rounds increases. Eventually, CG-PBFT can eliminate malicious nodes in the entire network within 6 consensus rounds. In comparison to PBFT, the CG-PBFT consensus algorithm can notably enhance the system network's security.

Meanwhile, this paper conducts quantitative experiments targeting specific security threat models, such as the Sybil attack model, replay attack model, and DoS attack model. To verify its security, the CG-PBFT algorithm is compared with traditional PBFT, CPBFT, and RBFT.

For the Sybil attack model, this paper designs the following experiment: 5%, 10%, and 15% Sybil nodes are injected into networks with 30, 50, and 80 nodes, respectively. The security is verified through 100 consensus rounds, and the experimental results are presented in Table 4.

Table 4: Witch node recognition rate comparison

	Witch node recognition rate		
Algorithm	**30, 5%**	**50, 10%**	**80, 15%**
PBFT	0	0	0
CPBFT	72.2 ± 2.3%	68.5 ± 1.8%	63.1 ± 1.9%
RBFT	79.4 ± 2.5%	75.1 ± 2.3%	70.8 ± 2.3%
CG-PBFT	98.7 ± 1.2%	97.2 ± 1.6%	95.6 ± 1.4%

Note: *Abbreviations*:30, 5% = 30 nodes, 5% malicious ratio; 50, 10% = 50 nodes, 10% malicious ratio; 80, 15% = 80 nodes, 15% malicious ratio.

According to the experimental results, CG-PBFT consistently maintains a high Sybil node identification rate through reputation score evaluation, node authentication, and grouping mechanisms, enabling it to effectively resist Sybil attacks.

For the replay attack model, this paper designs the following experiment: 30 legitimate identity authentication messages are intercepted, and an attacker is simulated to resend them. The experimental intervals are set to 1, 5, and 10 s, and the experiment is repeated 100 times to verify the defense effect of CG-PBFT. The experimental results are shown in Table 5.

Table 5: Interception success rate comparison

	Interception success rate		
Algorithm	**1 s**	**5 s**	**10 s**
PBFT	2.1 ± 0.3%	1.8 ± 0.2%	1.5 ± 0.2%
CPBFT	58.2 ± 1.8%	55.7 ± 1.6%	52.3 ± 1.5%
RBFT	73.4 ± 1.9%	70.1 ± 1.7%	67.8 ± 1.6%
CG-PBFT	98.7 ± 1.1%	98.4 ± 1.2%	97.9 ± 1.3%

From the experimental findings, it is evident that CG-PBFT achieves a higher interception success rate than the other three algorithms and can effectively intercept replay messages.

For the DoS attack model, this paper designs the following experiment: 100 nodes are divided into 5 groups, and 500 invalid consensus requests are sent per second to 20% of the nodes for 30 min. The experiment is repeated 100 times to verify the performance changes of CG-PBFT, with the experimental results presented in Table 6.

Table 6: Throughput retention rate and consensus delay increase rate comparison

Algorithm	**Throughput retention rate**	**Consensus delay increase rate**
PBFT	43.7 ± 2.2%	216.7 ± 3.4%
CPBFT	66.5 ± 1.9%	124.6 ± 2.4%
RBFT	72.4 ± 1.6%	98.3 ± 2.1%
CG-PBFT	92.5 ± 1.1%	37.8 ± 1.1%

As can be seen from the table, CG-PBFT can maintain a low consensus latency increase rate and a high throughput retention rate under DoS attacks, enabling it to effectively resist DoS attacks.

In summary, the CG-PBFT consensus algorithm can notably enhance the security of the system network.

5.5 Computational Cost Analysis

To confirm the practical feasibility of the algorithm, we strictly simulate the UAV embedded hardware environment (NVIDIA Jetson Xavier NX processor with 2 GB memory). Experiments are conducted 100 times under 30, 50, and 70 nodes to analyze its CPU utilization and memory usage, and the experimental results are displayed in Table 7.

According to the experimental results, the CPU utilization and memory usage of CG-PBFT are superior to those of the other three algorithms. It meets the requirements of the UAV embedded hardware environment and possesses certain practical feasibility.

Table 7: CPU utilization and memory usage comparison

	CPU utilization			Memory usage		
Algorithm	**30**	**50**	**70**	**30**	**50**	**70**
PBFT	58.3 ± 3.4%	67.9 ± 4.1%	76.5 ± 3.8%	628 ± 35 MB	815 ± 42 MB	986 ± 45 MB
CPBFT	52.1 ± 2.8%	60.5 ± 3.2%	68.8 ± 3.5%	612 ± 32 MB	798 ± 39 MB	965 ± 43 MB
RBFT	49.7 ± 2.6%	57.3 ± 3.0%	65.2 ± 3.3%	645 ± 34 MB	832 ± 41 MB	1012 ± 46 MB
CG-PBFT	32.5 ± 2.1%	38.7 ± 2.5%	45.3 ± 3.2%	386 ± 28 MB	458 ± 31 MB	512 ± 33 MB

6 Conclusion

Based on the task scenarios of large-scale UAV clusters, this paper tackles the limitations of PBFT, including excessive communication overhead, low throughput, and difficulty in eliminating malicious nodes. It thus devises a PBFT consensus algorithm named CG-PBFT, which integrates reputation scoring and a grouping mechanism. Firstly, CG-PBFT introduces a reputation scoring evaluation framework atop the original PBFT consensus algorithm. Experimental verification shows that it can maximize the reliability of selecting primary and leading nodes,diminish the count of malicious nodes within the network, and bolster the network system's security. Secondly, it employs group consensus methodologies and BLS aggregate signatures to optimize the consensus process and curtail communication complexity. Experimental outcomes reveal that, in comparison to the conventional PBFT consensus algorithm, CG-PBFT improves throughput, reduces consensus delay, and enhances the overall consensus efficiency. Finally, this paper indicates that CG-PBFT has a better effect than the traditional PBFT in the application scenarios of large-scale UAV clusters. However, the view change protocol of CG-PBFT still suffers from high latency when facing concentrated attacks by malicious nodes; moreover, it is currently only applicable to homogeneous UAV swarms, and its adaptability to heterogeneous nodes needs further optimization. In future research, we will optimize the view change mechanism and introduce predictive primary node election; extend it to the cross-cluster UAV consensus scenario; and integrate edge computing to reduce the local computational overhead of UAVs, thereby providing a more comprehensive solution for the consensus mechanism of distributed intelligent unmanned systems.

Acknowledgement: The authors are grateful to all the editors and anonymous reviewers for their comments and suggestions.

Funding Statement: This work was jointly supported by the following projects:Fund for technical areas of infrastructure strengthening plan projects under Grant 2023-JCJQ-JJ-0772.

Author Contributions: The authors confirm contribution to the paper as follows: study conception and design: Yuji Sang, Lijun Liu, Long Lv; data collection: Yuji Sang, Hemin Yin, Long Lv; analysis and interpretation of results: Yuji Sang, Husheng Wu, Long Lv; draft manuscript preparation: Yuji Sang, Lijun Liu, Long Lv. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: The data are contained within the article. And the authors confirm that the data supporting the findings of this study are available within the article.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest to report regarding the present study.

References

1. Zheng Z, Xie S, Dai HN, Chen X, Wang H. Blockchain challenges and opportunities: a survey. Int J Web Grid Serv. 2018;14(4):352–75. doi:10.1504/IJWGS.2018.095381.
2. Alsunaidi SJ, Alhaidari FA. A survey of consensus algorithms for blockchain technology. In: Proceedings of the 2019 International Conference on Computer and Information Sciences (ICCIS); 2019 Apr 3–4; Sakaka, Saudi Arabia. p. 1–6. doi:10.1109/ICCISci.2019.8716424.
3. Song J, Zhang P, Alkubati M, Bao Y, Yu G. Research advances on blockchain-as-a-service: architectures, applications and challenges. Digit Commun Netw. 2022;8(4):466–75. doi:10.1016/j.dcan.2021.02.001.
4. Alghamdi TA, Khalid R, Javaid N. A survey of blockchain based systems: scalability issues and solutions, applications and future challenges. IEEE Access. 2024;12:79626–51. doi:10.1109/ACCESS.2024.3408868.
5. Proof of stake [EB/OL]. [cited 2022 Sep 26]. Available from: https://en.bitcoin.it/wiki/Proof_of_Stake.
6. Saad SMS, Radzi RZRM. Comparative review of the blockchain consensus algorithm between proof of stake (POS) and delegated proof of stake (DPOS). Int J Innov Comput. 2020;10(2):27–32. doi:10.11113/IJIC.V10N2.272.
7. Xiong Y, Hu C. Comparative research on blockchain consensus algorithms. In: Proceedings of the 2022 International Confer-ence on Blockchain Technology and Information Security (ICBCTIS); 2022 Jul 15–17; Huaihua City, China. p. 160–4. doi:10.1109/ICBCTIS55569.2022.00045.
8. Yan S. Analysis on blockchain consensus mechanism based on proof of work and proof of stake. In: Proceedings of the 2022 International Conference on Data Analytics, Computing and Artificial Intelligence (ICDACAI); 2022 Aug 15–16; Zakopane, Poland. p. 464–7. doi:10.1109/ICDACAI57211.2022.00098.
9. Ferdous MS, Chowdhury MJM, Hoque MA. A survey of consensus algorithms in public blockchain systems for cryptocurrencies. J Netw Comput Appl. 2021;182:103035. doi:10.1016/j.jnca.2021.103035.
10. Pease M, Shostak R, Lamport L. Reaching agreement in the presence of faults. J ACM. 1980;27:228–34. doi:10.1145/322186.322188.
11. Lamport L, Shostak R, Pease M. The Byzantine generals problem. ACM Trans Program Lang Syst. 1982;4(3):382–401. doi:10.1145/357172.357176.
12. Castro M, Liskov B. Practical byzantine fault tolerance and proactive recovery. ACM Trans Comput Syst. 2002;20(4):398–461. doi:10.1145/571637.571640.
13. Li J, Jia YY, Zhang L. Practical Byzantine fault tolerance consensus algorithm of MuSig multi-signature. Appl Res Comput. 2025;42(2):352–6. doi:10.19734/j.issn.1001-3695.2024.06.0227.
14. Lao L, Dai X, Xiao B, Guo S. G-PBFT: a location-based and scalable consensus protocol for IoT-blockchain applications. In: Proceedings of the 2020 IEEE International Parallel and Distributed Processing Symposium (IPDPS); 2020 May 18–22; New Orleans, LA, USA. New York, NY, USA: IEEE; 2020. doi:10.1109/IPDPS47924.2020.00074.
15. Wang Y, Song Z, Cheng T. Improvement research of PBFT consensus algorithm based on credit. Singapore: Springer; 2019. doi:10.1007/978-981-15-2777-7_4.
16. Xu GX, Wang YS. Improved PBFT algorithm based on vague sets. Secur Commun Netw. 2022;2022:6144664. doi:10.1155/2022/6144664.
17. Yang J, Jia Z, Su R, Wu X, Qin J. Improved fault-tolerant consensus based on the PBFT algorithm. IEEE Access. 2022;10:30274–83. doi:10.1109/ACCESS.2022.3153701.
18. Cui Z, Xue F, Zhang S, Cai X, Cao Y, Zhang W, et al. A hybrid blockchain-based identity authentication scheme for multi-WSN. IEEE Trans Serv Comput. 2020;13(2):241–51. doi:10.1109/TSC.2020.2964537.
19. Raghav, Andola N, Venkatesan S, Verma S. PoEWAL: a lightweight consensus mechanism for blockchain in IoT. Pervasive Mob Comput. 2020;69(1):101291. doi:10.1016/j.pmcj.2020.101291.
20. Roman N, Vrabi R, Corn M, Požrl T, Diaci J. Distributed logistics platform based on blockchain and IoT. Procedia CIRP. 2019;81:826–31. doi:10.1016/j.procir.2019.03.207.
21. Qashlan A, Nanda P, He X, Mohanty M. Privacy-preserving mechanism in smart home using blockchain. IEEE Access. 2021;9:103651–69. doi:10.1109/ACCESS.2021.3098795.

22. Haque EU, Abbasi W, Almogren A, Choi J, Altameem A, Rehman AU, et al. Performance enhancement in blockchain based IoT data sharing using lightweight consensus algorithm. Sci Rep. 2024;14:26561. doi:10.1038/s41598-024-77706-x.
23. Kong L, Chen B, Hu F, Zhang J. Lightweight mutual authentication scheme enabled by stateless blockchain for UAV networks. Secur Commun Netw. 2022;2022:2330052. doi:10.1155/2022/2330052.
24. Zhang L. A cross network identity authentication scheme for UAVs based on layered blockchain technology. In: Advances in intelligent networking and collaborative systems. Cham, Switzerland: Springer; 2024. doi:10.1007/978-3-031-72322-3_13.
25. Boneh D, Lynn B, Shacham H. Short signatures from the weil pairing. In: International Conference on the Theory and Application of Cryptology And Information Security. Berlin/Heidelberg, Germany: Springer; 2001. p. 514–32. doi:10.1007/3-540-45682-1_30.
26. Zhu W, Shi L, Li J, Cao B, Wei K, Wang Z, et al. Trustworthy blockchain-assisted federated learning: decentralized reputation management and performance optimization. IEEE Internet Things J. 2025;12(3):2890–905. doi:10.1109/JIOT.2024.3480995.
27. Ye Z, Wen T, Liu Z, Song X, Fu C. An efficient dynamic trust evaluation model for wireless sensor networks. J Sens. 2017;2017:7864671. doi:10.1155/2017/7864671.