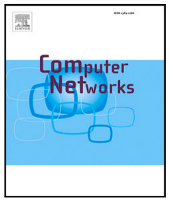
Computer Networks 251 (2024) 110612

Contents lists available at ScienceDirect

Computer Networks

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



Leveraging lightweight blockchain for secure collaborative computing in UAV Ad-Hoc Networks

Runqun Xiong [a,*], Qing Xiao [a], Zhoujie Wang [b], Zhuqing Xu [a], Feng Shan [a]

[a] School of Computer Science and Engineering, Southeast University, Nanjing, 211189, PR China
[b] Tencent Technology (Shanghai) Co.Ltd., Shanghai, PR China

ARTICLE INFO

Keywords:
UAV Ad-Hoc Network
Blockchain
Collaborative computing
Consensus
Smart contract

ABSTRACT

Unmanned Aerial Vehicle (UAV) Ad-Hoc Networks (UANET) enable collaborative work among UAVs for versatile task execution, but they face security challenges due to physical vulnerabilities, software issues, and dynamic wireless networks. This paper proposes a secure collaborative computing framework based on blockchain technology. Specifically, we first design a lightweight blockchain scheme suitable for UANET and present an improved Practical Byzantine Fault Tolerance (PBFT) consensus algorithm based on trust evaluation, aiming to reduce consensus overhead and establish trust relationships among UAVs. Furthermore, we devise a smart contract-based UAV task allocation strategy that considers both task execution efficiency and offloading security. This strategy enables UAVs to make optimal task-offloading decisions and facilitates collaborative computing according to smart contract rules. Simulation results demonstrate that our proposed consensus algorithm reduces average consensus latency by 47% and message count by 76% compared to PBFT-based algorithms. Additionally, the task allocation strategy decreases task costs by 48% compared to local computing strategies, achieving efficient and secure collaboration among UAVs in UANET. The real-world experiments with a UAV swarm further validate the efficiency and security of our framework, confirming its practical applicability in UANET.

1. Introduction

Recently, unmanned aerial vehicles (UAVs) have been widely adopted for applications like infrastructure inspection and search-and-rescue [1–3] due to their small size, versatility, flexibility, and low operating costs. However, their limited onboard capabilities pose challenges for compute-intensive tasks. To address this, multiple UAVs can establish temporary aerial networks called UAV Ad-Hoc Networks (UANETs) [4] to collaborate and execute assigned tasks autonomously. UANETs also facilitate data collection, sharing and processing, especially in remote, complex terrains with disrupted infrastructure where networks are unavailable. For instance, consider a disaster response scenario where a fleet of 100 UAVs is deployed by multiple organizations to perform tasks such as real-time mapping, search and rescue operations, and environmental monitoring. These UAVs have varying computational capacities due to differences in onboard hardware. The need for timely processing of large amounts of data, such as high-resolution imagery, necessitates offloading computationally expensive tasks to UAVs with available idle capacity. Task offloading in this scenario is necessitated by the limited onboard computing capabilities of individual UAVs in the UANET. Given constrained resources and power-intensive nature of flight, UAVs often lack the computational power to efficiently execute complex tasks such as real-time data processing, image recognition, or computational simulations, which are essential for a wide range of applications including infrastructure inspection, search-and-rescue operations, and so on.

Additionally, with increasing UANET adoption for performing commissions, data integrity and communication security among collaborating UAVs are critical concerns [5]. For example, a UANET used to track drug traffickers near borders may face wiretaps and attacks due to open links and dynamic topologies blanketing mission-critical areas. In practice, given the vulnerability of flying UAVs and limited onboard batteries, single UAV continuous operation time is constrained [6]. To enable long-term operation in target areas, UANETs must recycle low-battery nodes and add new nodes, requiring urgent capabilities for authenticating and tolerating UAV replacement [7]. This makes UANETs vulnerable to control signal spoofing attacks that transmit false signals or even seize control via successful cyber-attacks, posing safety threats [8]. Additionally, lacking fixed infrastructure, many multi-UAV systems rely on central nodes or edge servers to store and process data, risking single points of failure [9]. UAVs may also fail communicating

* Corresponding author.
E-mail address: rxiong@seu.edu.cn (R. Xiong).

https://doi.org/10.1016/j.comnet.2024.110612
Received 12 December 2023; Received in revised form 21 June 2024; Accepted 21 June 2024
Available online 25 June 2024

with control centers in real time when links are unavailable, especially during emergencies. Thus, enhancing UANET security is essential for such scenarios.

To prevent privacy leakage and ensure data integrity in UANETs, blockchain as a decentralized solution is expected to securely and adaptively maintain credible collaborative computing among dynamic UAV nodes [5]. UAV wireless broadcasts match blockchain principles. When a UAV produces and broadcasts blocks/transactions, neighboring nodes can quickly receive and verify messages, more efficient than wired blockchain networks with centralized gateways and bottlenecks. However, directly deploying traditional blockchains on resource-constrained UANETs is challenging due to power-intensity, low throughput, and centralization trends. For instance, each UAV would store ever-growing blocks, incurring major storage overhead. Acquiring unique hashes per block via common Proof-of-Work (PoW) consensus requires substantial computing resources [10]. Delegated-Proof-of-Stake (DPoS) increases efficiency by reducing consensus nodes but sacrifices decentralization [11]. Practical Byzantine Fault Tolerance (PBFT) ensures correct consensus given less than one-third malicious nodes [12] but performs optimally only in small fixed-size networks, which may not satisfy demands of large-scale dynamic UANETs. Thus, customized decentralized solutions are needed for resource-constrained UANETs.

To address the aforementioned challenges, we propose *AerialBC*, a secure collaborative computing framework for resource-constrained UANETs. Unlike existing approaches, *AerialBC* incorporates a lightweight blockchain scheme tailored for UANET environments, which tackles issues related to power-intensity, low throughput, and centralization trends inherent to traditional blockchain solutions. The lightweight nature of our blockchain is achieved through reduced computational and storage requirements, which is crucial for resource-constrained UAVs. By optimizing consensus algorithms and minimizing the need for extensive data storage, our approach ensures efficient and secure operations within UANET environments. Furthermore, we propose an improved trust-enhanced PBFT consensus algorithm that not only reduces consensus overhead but also establishes dynamic trust relationships among UAVs, which is critical in highly mobile and dynamic UANETs. Additionally, the smart contract-based task allocation strategy introduced in this work is distinct in its consideration of both task execution efficiency and offloading security, enabling UAVs to make optimal decisions based on smart contract rules, a feature that is not available in existing literature. We highlight the following key contributions:

- We propose a secure collaborative computing framework named *AerialBC* for resource-constrained UANET based on a lightweight blockchain scheme.
- We design an improved trust-enhanced PBFT consensus mechanism to build a lightweight blockchain solution tailored for UANET.
- We develop a smart contract-based UANET task allocation strategy to enhance the efficiency of collaborative computing for *AerialBC* and ensure system security in the face of risks.
- We validate the performance of *AerialBC* through simulations and real-time experiments, revealing its ability to detect malicious nodes in UANET and enhance the security of collaborative computing in UANET while enabling efficient task execution.

2. Related work

Lately, many studies have explored integrating blockchain technology with Vehicular Ad-hoc Network (VANET) and UANET applications. Guo et al. [13] proposed an attribute-based data sharing scheme using blockchain for 6G-enabled VANETs to enhance data security and sharing efficiency. Xie et al. [14] introduced AirCon, an over-the-air consensus mechanism for wireless blockchain networks, which significantly reduces communication overhead and latency. Chen et al. [15] developed a vehicular trust blockchain framework with scalable Byzantine consensus, addressing scalability and trust management in vehicular networks. Su et al. [16] devised a lightweight vehicular blockchain framework for UAV-assisted IoV. They utilized a credit-based consensus algorithm to improve efficiency and security. Abegaz et al. [17] proposed a framework combining multi-agent deep reinforcement learning, blockchain, and game theory to manage resource trading in multi-UAV networks. Xu et al. [18] introduced blockchain for UAV-assisted IoT data collection and incentivized UAVs with charging tokens. However, they focused on UAVs' supportive role without considering UAV swarm cooperation security [19–22]. Ge et al. [23] proposed a distributed UAV blockchain scheme to address security threats while minimizing overhead. Gupta et al. [24] presented a blockchain-based secure UAV communication scheme over 6G. Abichandani et al. [25] enabled secure UAV data sharing through smart contracts. Gai et al. [26] introduced blockchain for identity authentication in UAV networks.

Several works have studied UAV-based edge computing. Liu et al. [27] proposed a joint optimization approach for workflow assignment and routing in a UAV-edge-cloud model to minimize cost and latency. Messous et al. [28] developed a game-theoretic model for UAV swarm offloading decisions balancing local processing and offloading tradeoffs. Kang et al. [29] utilized UAVs for edge computing resource allocation to maximize completed tasks under quality of service constraints. Callegaro et al. [30] addressed the UAV offloading decision problem considering task latency and energy. Xu et al. [31] proposed a blockchain-based resource pricing and trading scheme between edge servers and UAVs using Stackelberg game theory. In [32], the authors investigated UAV-based multi-access edge computing, decomposing the cost minimization problem into stochastic games for offloading and server deployment with learning algorithms to achieve polynomial time complexity. Some studies have explored UAV-to-UAV offloading to enhance resource utilization in multi-UAV scenarios, given the limited coverage of edge/cloud servers compared to highly dynamic UAVs. Mukherjee et al. [33] optimize multi-hop paths for computational offloading in a UAV swarm to minimize energy consumption. Liu et al. [34] study computation offloading from user UAVs to edge nodes under latency constraints, optimizing offloading targets, channels, and rates to reduce energy consumption. In [35], the authors jointly optimize real-time UAV relay deployment and resource allocation for disaster rescue. Gao et al. [36] propose a deep reinforcement learning-based task assignment method for UAV-based mobile crowdsensing to optimize sensing coverage and data quality. Ouahouah et al. [37] study task offloading from UAV cluster heads to members.

In summary, previous studies have explored blockchain applications in UAV-assisted scenarios, focusing on resource management, data collection, and communication protocols. However, they have not addressed the unique challenges of UANETs, such as resource constraints and dynamic topology. Our improved PBFT algorithm integrates trust mechanisms within the consensus process, and our smart contract-based strategy tackles task allocation from a security perspective. *AerialBC* contributes to the literature by presenting a lightweight blockchain-based secure collaborative computing framework specifically designed for UANET, optimizing efficiency, security, and trust management within a cohesive system.

3. System model

In this section, we introduce the system model of *AerialBC*, a secure collaboration framework for resource-constrained UANET based on a lightweight blockchain.

3.1. Overview of AerialBC

As shown in Fig. 1, *AerialBC* comprises a UAV swarm, a ground station, and a blockchain network. The UAV swarm forms a UANET in

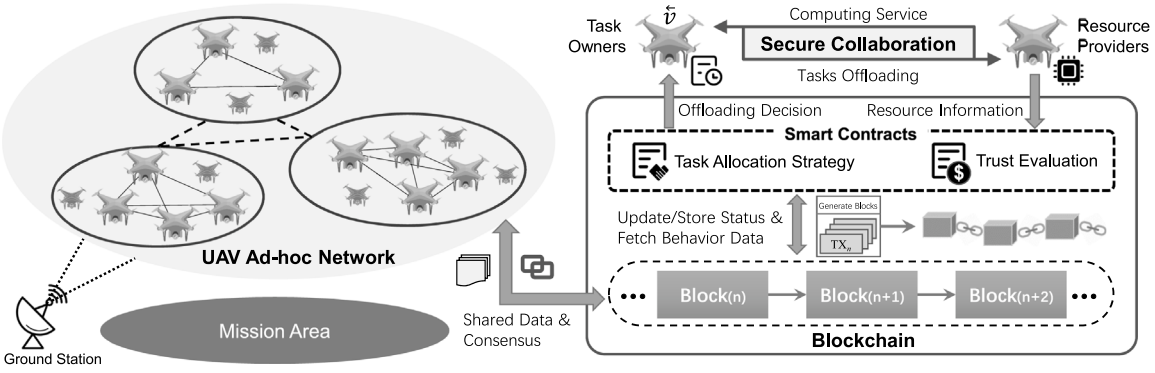


Fig. 1. System model of AerialBC.

the mission area, offering data collection and network communication to ground sensors or devices. The blockchain is deployed on UANET nodes, functioning as a distributed ledger that records the behavior of the UAVs during task execution and collaborative computing processes. This behavior serves as input for trust evaluation, determining the trustworthiness of each node in the UANET. The trustworthiness supports the selection of the committee in the consensus algorithm and the trust mechanism in collaborative computing. Subsequently, the participating UAVs upload task and resource information to the blockchain for collaborative computing. The task allocation strategy, deployed on the blockchain's smart contract, makes task offloading decisions for the participating UAVs based on time cost, and trustworthiness while ensuring system efficiency and security requirements are met. The UAVs adhere to the smart contract rules to complete the task offloading process, ultimately aiming to enhance system performance, security, and reliability.

3.2. Network model

In the collaborative computing of UANET, the primary entities encompass a UAV swarm, ground station, and blockchain network.

UAV swarm. As shown in Fig. 1, a swarm of UAVs $\mathcal{N} = \{1, \ldots, n, \ldots, N\}$ is deployed to form UANET. Each UAV serves a dual purpose: firstly, it functions as a UANET node and blockchain node, actively participating in consensus and maintaining the integrity of the blockchain network. Secondly, it acts as an aerial network access point, gathering sensor or environmental data from the mission area and providing data services to ground devices. Within the swarm of UANET, UAVs can be distinguished based on their computing capabilities, which are quantified by their computational capacity (measured in CPU cycles per second). High-computing UAVs are those with higher computational capacities and typically have more advanced processors or dedicated hardware that enables them to perform compute-intensive tasks more efficiently. In contrast, low-computing UAVs have lower computational capacities and are usually equipped with less powerful processors, limiting their ability to execute complex computations quickly. To systematically identify high-computing and low-computing UAVs within the network, we employ a registration process where each UAV declares its computational capacity upon joining the network. This information is recorded and maintained by the ground station or a designated master node in the UANET blockchain. The computational capacity data is then used in the task allocation process to optimally offload tasks from task owner UAVs to resource provider UAVs based on their computing capabilities, thus ensuring efficient utilization of computational resources across the swarm.

UAV Ad hoc network. Massive UAVs can self-organize as a temporary distributed ad hoc network $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, consisting of vertices (UAVs) $\mathcal{V}$ and edges (wireless links) $\mathcal{E}$. UAVs are connected to each other through UAV-to-UAV (U2U) links to greatly increase the coverage of the swarm. Each UAV can broadcast its messages to other UAVs through the wireless channel.

Ground station. Ground station (GS) serves as the control center of the UAV swarm with the highest authority. GS has the capability to issue certificates for the admission of new UAVs into the network. In our system, GS communicates with the swarm only at the beginning through GS-to-UAV (G2U) links. However, UAVs may fail to transmit data to the GS in real time due to no available backhaul links, especially in emergencies. No one node can obtain multiple identities at the same time.

Blockchain network. The blockchain is a distributed ledger system with admission nodes and shared blocks. It can be deployed within a UANET to create a secure, aerial wireless, private blockchain network. UAV transactions on this blockchain are securely recorded on an immutable and traceable ledger which supporting the trust evaluation of UAVs. Furthermore, blockchain-based smart contracts, serving as trusted computing tools, facilitate task allocation strategies and standardize UANET collaboration process. In our work, the transactions stored by UAVs in the blockchain include task-related transactions, consensus participation records, and behavioral alerts. Task-related transactions encompass task declarations, task allocations, and task completion confirmations. These transactions create an immutable record of the task lifecycle within the UANET, allowing for traceability and dispute resolution. Consensus participation records include votes and decisions made during the consensus process, which contribute to the transparency and reliability of the network's decision-making process. Behavioral alerts are transactions that report potential misbehavior or malicious activities by UAVs, which are crucial for maintaining network security and trustworthiness.

3.3. Computation model

In the scenario of UANET cooperation in computational tasks, as depicted in Fig. 1, due to the indivisibility of the computation tasks, high-load UAV nodes have two options: either locally execute the tasks or offload them to other UAVs for execution. Considering the stochastic arrival and diverse nature of tasks, different UAVs have varying demands for computational power at different times, indicating whether they are providers or consumers of computing resources. Based on this, we classify the nodes into Task Owner UAVs (TO), Resources Provider UAVs (RP), and Relay Node UAVs (RN) that serve as relays for data transmission in the UANET. Let us assume there are a total of J resource provider UAVs in UANET, which can be represented by the set $\mathcal{RP} = \{RP_j | j \in \{1, 2, \ldots, J\}\}$. Here, RP_j denotes the resource provider UAV with index j. Similarly, let us assume there are I task owner UAVs, represented by the set $\mathcal{TO} = \{TO_i | i \in \{1, 2, \ldots, I\}\}$. Here, TO_i represents the task owner UAV with index i. During a task assignment, each UAV submits a task requirement. If the TO_i has a computational task that needs to be processed, this task can be represented by a triplet:

$task_i = \{s_i, \kappa_i, t_i^{\max}\}$, where $i \in \{1, 2, \ldots, I\}$. In this representation, s_i denotes the original data size of the computational task, κ_i represents the computational efficiency of the task, *i.e.*, the number of CPU cycles required per bit of data computation, and $t_i^{\max}$ indicates the time sensitivity or desired completion time of TO_i for the $task_i$.

There are two computation modes for $task_i$ of TO_i: the local mode, where the task is executed locally using the idle computing power available on TO_i; and the offloading mode, where the task is transferred over UANET to RP_j with higher idle computing power. We introduce binary variables $\{x_{ij} | j \in \{1, 2, \ldots, J\}\}$ and x_i^{local} to represent the selected computation mode for $task_i$, whether it is offloading or local mode, as well as the identifier of the resource provider UAV in case of offloading mode. If the local mode is chosen, then $x_i^{local} = 1$ and all x_{ij} variables are set to 0. If the offloading mode is chosen and the selected resource provider UAV is RP_j, then $x_{ij} = 1$ and all other binary variables are set to 0.

3.4. Threat model

The following attack types are considered in *AerialBC*:

Single point of failure. A single point of failure occurs when a single node failure results in the entire system becoming paralyzed. This is unacceptable for UANET, which prioritizes high availability. Because of the vulnerability of UANET nodes and their limited flight time due to battery constraints, the probability of node replacement or single-point failure is high, making single-point of failure a significant security threat to UANET.

False task attack. UAVs, lacking physical security or having software vulnerabilities, are susceptible to being attacked and turned into malicious nodes. Malicious nodes may intentionally issue false tasks to other nodes in order to waste their computational resources and energy. This type of attack leads to resource wastage, task execution delays, and degradation of system performance.

Active malicious nodes. In many IoT systems, normal nodes tend to go into sleep mode to conserve energy when they have no tasks to perform. This results in their low level of participation in the network. Active malicious nodes exploit this by remaining active and attempting to gain control over the network. These nodes may engage in activities such as continuously broadcasting false information, interfering with consensus processes, or attempting to monopolize network resources. By staying active and appearing more reliable than the dormant normal nodes, active malicious nodes can gradually increase their influence and control over the network.

4. Improved trust-enhanced PBFT consensus

In this section, we propose an improved trust-based PBFT consensus for the UANET blockchain. First, we design a trust evaluation based on UAV behavior records that considers swarm activity levels and working characteristics, effectively assessing reliability and establishing trust. Second, by selecting high-trust UAVs to execute consensus, we significantly improve efficiency, reduce overhead, and enable blockchain on resource-constrained UAVs. Moreover, committee cycles handle dynamic join/departure, adapting to node changes due to power or faults.

4.1. Trust evaluation

In *AerialBC*, each UAV's trustworthiness $Trust_i$ is stored in the trust ledger within block headers. It varies as behaviors are recorded on-chain. UAV behavior records are composed of entries that log significant actions taken by UAVs within the network. These actions include, but are not limited to, successful task completions, participation in consensus rounds, timely response to task allocations, and adherence to flight and safety protocols. Positive behaviors like block packaging increase trust, while negatives like consensus failures decrease it. To detect malicious behavior, each UAV monitors the actions of others based on predefined security rules and operational benchmarks. For instance, a UAV may be flagged as acting maliciously if it consistently fails to complete assigned tasks, does not participate in consensus or voting processes, attempts to monopolize resources, or deviates from prescribed flight paths without valid reasons. When a UAV observes a potential breach of protocol or malicious intent, it can initiate a suspicion transaction by creating an accusation contract. This transaction is then broadcasted to the network, prompting other UAVs to vote on the proposal based on their observations and the evidence provided. If the accusation is validated through consensus, the trust score of the accused UAV is adjusted accordingly, and the misconduct is recorded on the blockchain. Inspired by [38], we divide $Trust_i$ into two components:

$$Trust_i = \lambda_1 Trust_i^P - \lambda_2 Trust_i^N, \tag{1}$$

where, $Trust_i^P$ denotes the positive behavior and $Trust_i^N$ represents the negative behavior. The weight coefficients for the two components are denoted as λ_1 and λ_2 respectively. The negative part $Trust_i^N$ can be denoted as:

$$Trust_i^N = \min\left(\sum_{k=0}^{neg_i} \frac{f_\Theta(\Theta_k)}{neg_i - k + 1}, Trust_{max}^N\right), \tag{2}$$

where neg_i represents the number of negative behaviors of UAV_i, Θ_k represents the negative behavior with index k, f_Θ represents the function to decide the punishment of each negative behavior. The influence of negative behaviors fades with index rather than time, and the most recent negative behavior will assume more influence. Once $Trust_i^N$ reaches $Trust_{max}^N$, the upper bound of the negative part, UAV_i will be expelled. To better access the impact for the reward from the positive behaviors, we define idx_{ng} is the recent closest index of negative behaviors. And $z = idx_{ng} + 1$, is a resetting variable used to keep track of the starting point after each negative behavior. Meanwhile, the positive part $Trust_i^P$ is defined as:

$$Trust_i^P = \min\left(\sum_{k'=z}^{pos_i} \frac{f_\Phi(\Phi_{k'})}{pos_i - k' + 1}, Trust_{max}^P\right), \tag{3}$$

where pos_i represents the number of positive behaviors of UAV_i, $\Phi_{k'}$ represents the positive behavior with index k', f_Φ represents the function to decide the reward of each positive behavior, $Trust_{max}^P$ is the upper bound of $Trust_i^P$ to avoid unlimited accumulation of the positive behaviors from the sequential activities, z represents the index of the first positive behavior after the most recent negative behavior. Accumulating rewards from z will drive UAVs to perform consecutive positive behaviors.

The $Trust_i^P$ and $Trust_i^N$ values which initialized as a fixed positive values are recorded by each node through transactions. These values are finally summarized and computed by the consensus node to obtain the trust values which we demonstrate detailedly in the following section. When the records are written into the block, the consensus for the final trust values is reached. Thus, the blockchain maintains historical trust values.

4.2. Consensus process

(1) Registration and Initialization. During initialization, ground station (GS) grants a pair of keys to each UAV_i, including the public key PK_i and the secret key SK_i. Also, a certificate CTF_i is granted, which can be regarded as the result of encrypting PK_i with the secret key of GS. As shown in Fig. 2, in the ***Enc*** function, the data is encrypted using the private key of the data receiver and *nonce* means a random number, while in the ***Dec*** function, the corresponding public key is used for decryption, since the public key of GS, denoted as PK_{GS}, has been hard-coded into UAVs, UAV_x can verify the certificate from UAV_i and decide whether it is permitted. In this process, private blockchain is formed with permitted UAVs.

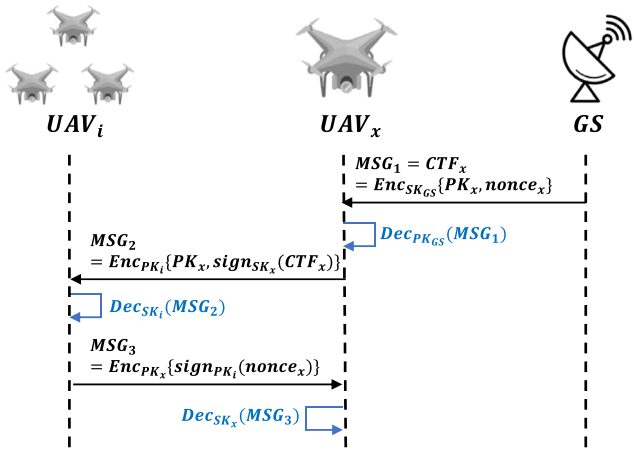


Fig. 2. Process of authentication for a newly added UAV.

The GS serves as the Certificate Authority (CA) role in our system and the network can be seems as lightweight permissioned blockchain network.

(2) Transaction Generation and Broadcasting. Transactions in the blockchain network are categorized into three types: regular transactions in collaborative computing, suspicious transactions regarding improper behavior of UAVs, and special transactions for modifying the current consensus configuration (*e.g.*, committee updates, view switches).

When UAV_i observes suspicious behavior by another UAV, it has the capability to send a indicating distrust transaction to accuse the target of suspicion, represented as transaction tx_s in the following format:

$$tx_s = \langle PK_i, PK_j, time, location, Evidence, Sig_i \rangle . \tag{4}$$

Here, the proposal contains the public keys of both parties, the time and location of suspected node misconduct, evidence obtained through communication or sensors ($Evidence$), and the signature of the suspected node (Sig_i).

When a suspicious transaction is broadcasted in the UANET, other UAVs vote on the proposal based on actual observation. If the proposal receives enough votes according to subsequent weighted voting mechanisms, it will be included in a block by the committee. Consequently, the suspected UAV is recorded with one negative behavior, leading to a decrease in trustworthiness.

(3) Miner Selection. In traditional PBFT, each node exchanges transactions with all others for consensus, incurring high communication demand that limits scalability. Our key idea is that since UAV trustworthiness is recorded, we generate a small committee. Specifically, top M trusted UAVs are selected as miners $\mathcal{M} = \{1, \ldots, m, \ldots, M\}$, $M \leq N$. Other UAVs are observers. Only miners perform transaction/block processing and validation by taking turns as the primary. Consensus switches primaries regularly and upon failures(called a view change). Thus, consensus is only among miners, reducing consumption. Observers propose transactions but do not participate. Moreover, dynamic miner/observer roles prevent attackers from easily controlling the network.

UAVs enter/leave the blockchain dynamically due to factors like battery/link failures, with varying trust over time. For scalability and security, the committee is updated regularly (*e.g.* every 40 views) during stage changes that temporarily refuse transactions. Since the trust ledger is locally stored, *Aerial BC*'s stage change is faster and cheaper than voting schemes based on DPoS. Only consensus members and transaction senders need recording rather than all nodes continuously, reducing complexity.

(4) Block Generation. Blocks are produced by miners. We do not directly take the ranking of trustworthiness as the view change sequence, which makes the trustworthiness of the later phase significantly lower than the early phase during a stage. Malicious UAVs tend to attack the later phase. We rearrange the order of producers to make the trustworthiness distribute more evenly, which can prevent attacks against consecutive blocks.

4.3. Proposal voting scheme

In *Aerial BC*, ensuring precise proposal voting and behavior recording necessitates the consideration of UAV trustworthiness. This involves assigning distinct weights to individual UAVs based on their trust levels. Proposal outcomes should integrate both trust values and feedback. We represent the weight of UAV_i as:

$$w_i = \max(Trust_i - Trust^N_{max}, 0). \tag{5}$$

Thus, the voting result for a p can be denoted as:

$$L_{p,r} = \begin{cases} 1, & \text{if } \dfrac{\sum_{k=1}^{N} w_k \cdot (vote_p(k) + 1)}{2 \cdot \sum_{k=1}^{N} w_k} \geq r, \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

where r represents the required level for the swarm to approve p and its value varies in $[0.5, 1]$. Proposals with high significance should be appointed a higher r. $vote_p(k)$ represents the feedback from UAV_k for proposal p, which values in $\{-1, 0, 1\}$. The proposal is approved when $L_{p,r} = 1$.

5. Smart contract-based task allocation strategy

In this section, we design a task allocation strategy considering both efficiency and security of UANET. Previous works [34,39] concentrated tasks on resource-rich UAVs to improve utilization and efficiency. However, they overlooked UAVs' vulnerabilities like physical fragility and limited energy, leading to reliability threats from malicious fake task attacks. In contrast, our strategy considers operational costs like transmission and execution costs, as well as risk costs in unstable environments. It promotes balanced allocation using load metrics to enhance robustness. It also incorporates node trustworthiness from Section 4 to measure false task attack risks. This strategy is deployed on blockchain smart contracts, providing a standardized collaborative computing process in UANET through two-stage task offloading. The objective is facilitating secure and efficient UAV cooperation. However, upcoming routine or unplanned UAV replacements will impact ongoing tasks. Key notations are shown in Table 1.

5.1. Task cost

In order to optimize the task allocation strategy of our *Aerial BC*, a modeling analysis is conducted on the operational costs and risk costs under two different computational approaches for the task owner UAVs. Additionally, considering time as a crucial factor affecting system performance and efficiency (*e.g.*, task completion time, transmission time), the modeling primarily revolves around time in terms of operational costs.

5.1.1. Local computing mode

(a) Task Computing Time. If the task owner UAV TO_i decides to adopt the local computing mode, *i.e.*, TO_i will execute the task $task_i = \{s_i, \kappa_i, t_i^{\max}\}$ locally. In this mode, the task duration is equal to the execution time of the task locally. Therefore, the total duration in the local mode t_i^{local} can be represented as:

$$t_i^{local} = t_i^{com}, \tag{7}$$

Table 1
The key notations.

Symbol	Description
TO_i	Task owner UAV with index i
RP_j	Resource provider UAV with index j
$task_i$	The task assigned to UAV_i
s_i	The original data size of $task_i$
κ_i	CPU cycles required per unit data for $task_i$
$t_i^{\max}$	Desired completion time for $task_i$
f_i^{local}	Remaining local computational capacity of UAV_i
π_i	Computational workload per unit desired completion time for $task_i$
Δ_{ij}	Risk factor associated with offloading a task from TO_i to RP_j, dependent on UAV node trustworthiness
ξ	Time cost coefficient
x_{ij}	Offloading decision for $task_i$, where $x_{ij} = 1$ indicates offloading task from TO_i to RP_j
x_i^{local}	Offloading decision for $task_i$, where $x_i^{local} = 1$ indicates local computation by TO_i
t_{ij}^{d2d}	Transmission time for offloading task from TO_i to RP_j
t_{ij}^{com}	Execution time for offloading task from TO_i to RP_j
f_j^{cap}	Idle computational capacity of RP_j in the current collaborative computing session
f_{ij}	Computational capacity allocated by RP_j to TO_i in the current collaborative computing session
$Cost_{TO_i}$	Total cost for TO_i
p_j	Load balancing metric for RP_j

where, t_i^{com} represents the computation time of the task locally. Assuming TO_i has local idle computing power f_i^{local}, the execution time locally can be expressed as:

$$t_i^{com} = \kappa_i s_i (f_i^{local})^{-1}. \tag{8}$$

(b) Cost Function. The cost of TO_i in local computing mode primarily consists of the time cost incurred by task computations. Let ξ be the coefficient representing the cost of tasks per unit of time. The total cost $Cost_{TO_i}^{local}$ of TO_i in local mode can be expressed as:

$$Cost_{TO_i}^{local} = \xi t_i^{local} x_i^{local} \tag{9}$$

where, $x_i^{local} = 1$ indicates that TO_i selects the local computing mode.

5.1.2. Offloading computing mode

In the offloading computing mode, assuming TO_i offloads its task to RP_j for computation, the time cost of transferring the task from TO_i to RP_j and the computation time cost on RP_j collectively contribute to the operational cost of the system. So, the completion time t_{ij}^{edge} in the offloading mode is represented as the sum of three processes: task transfer time t_{ij}^{d2d}, task execution time t_{ij}^{com}, and result feedback time t_{ij}^{res}. It can be expressed as follows:

$$t_{ij}^{edge} = t_{ij}^{d2d} + t_{ij}^{com} + t_{ij}^{res}. \tag{10}$$

Considering that the data size of the feedback results is typically small, the term t^{res} can be neglected. Furthermore, since the UAV node always remains within the UANET during operation, there will not be a situation where there is no feedback link due to node movement.

(a) Task Transfer Time. Due to unstable wireless mobility of UAV nodes, the dynamic status of network topology and link make it challenging to establish an end-to-end multi-hop forwarding communication model. In practice, IEEE 802.11 ad hoc networks in UANETs require routing protocols like OLSR for multi-hop transmission support. OLSR is currently the most widely used UANET routing protocol. Complex environmental factors influence wireless links between UAV nodes, resulting in fluctuating packet loss, delay, and bandwidth. Evaluating links solely on hop count is infeasible. We propose utilizing OLSR's link evaluation to assess highly dynamic communication states and calculate multi-hop delays. OLSR assesses link quality by sending HELLO packets to measure reception/transmission conditions. Received link quality (LQ) and transmitted neighbor link quality (NLQ) indicate historical packet loss rates. Higher LQ/NLQ corresponds to lower link cost (ETX) - the expected transmissions needed to successfully transmit a packet. However, LQ and NLQ have latency. Also, UAV mobility causes frequent link fluctuations. To address this, we incorporate node positions, motion directions, and neighbor information into link evaluation. The ETX calculation is:

$$ETX(\eta) = e^{\upsilon_\eta \gamma}(\phi(\eta)\rho(\eta))^{-1}, \tag{11}$$

where η is a link, $\phi(\eta)$ and $\rho(\eta)$ are the NLQ and LQ of η, υ_η is the relative motion speed of connected nodes through η, and γ is a non-negative coefficient. The relative speed is calculated and maintained by propagating position and motion data in OLSR packets. In UANET, end-to-end paths comprise multiple links, with path cost as the link cost sum, $ETX(R) = \sum_{\eta \in R} ETX(\eta)$, where R is the minimum-cost end-to-end path and η are its links. If $\omega(\eta)$ is the basic transmission rate of link η, the multi-hop end-to-end rate approximates the ratio of basic rate to link cost. For task i of owner i defined as $\text{task}_i = \{s_i, \kappa_i, t_i^{max}\}$ with data size s_i, its estimated transmission time t_{ij}^{d2d} to provider j is:

$$t_{ij}^{d2d} = \sum_{\eta \in R_{ij}} t_\eta^{d2d} = s_i \sum_{\eta \in R_{ij}} \frac{ETX(\eta)}{\omega(\eta)}. \tag{12}$$

Let $\varpi_{ij} = \sum_{\eta \in R_{ij}} \frac{ETX(\eta)}{\omega(\eta)}$ be the sum of the ETX-weighted values for each η in R_{ij}. If s_i represents the data transmission amount, then t_{ij}^{d2d} can be expressed as:

$$t_{ij}^{d2d} = s_i \varpi_{ij}. \tag{13}$$

(b) Task Computing Time. After the task is transmitted from TO_i to RP_j, it undergoes computation at RP_j. Let f_{ij} denote the amount of computational resources allocated by RP_j from its available idle computing power to TO_i, which corresponds to the number of CPU cycles per second that can be utilized. Since the total CPU cycles required for the computation task $task_i = \{s_i, \kappa_i, t_i^{\max}\}$ is $\kappa_i s_i$, the expected execution time of the task at RP_j, denoted as t_{ij}^{com}, can be expressed as:

$$t_{ij}^{com} = \kappa_i s_i (f_{ij})^{-1}. \tag{14}$$

For RP_j, the available idle computing resources it can provide are limited. Let f_j^{cap} represent the upper limit of idle computing power that RP_j can offer in this collaborative computation. Therefore, for all UAVs that choose to offload their tasks to RP_j, the total computing power provided by RP_j should not exceed its own computing capacity, *i.e.*,

$$\sum_{i=1}^{I} f_{ij} x_{ij} \le f_j^{cap}. \tag{15}$$

Assuming that RP_j allocates tasks using its maximum computing power and distributes the computational resources proportionally when receiving multiple tasks. Considering that UANET is a time-sensitive network and the computational workload is also a crucial factor in task allocation, we employ a weighted allocation method based on the unit expected computation time. It can be expressed as follows:

$$f_{ij} = f_j^{cap} \frac{\frac{\kappa_i s_i}{t_i^{\max}}}{\sum_{k=1}^{I} x_{kj} \frac{\kappa_k s_k}{t_k^{\max}}}. \tag{16}$$

Here, $\frac{\kappa_i s_i}{t_i^{\max}}$ serves as a weighted indicator, representing the computational workload of a task per unit expected time. Under this allocation method, tasks with higher computational workloads per unit expected time are assigned greater weights, which helps improve the efficiency of completing time-sensitive tasks. Let $\pi_i = \frac{x_{ij}\kappa_i s_i}{t_i^{\max}}$, then the expected computation execution time of the task at RP_j can be expressed as:

$$t_{ij}^{com} = \frac{\kappa_i s_i \sum_{k=1}^{I} \pi_k}{\pi_i x_{ij} f_j^{cap}}. \tag{17}$$

(c) Cost Function. In addition to the aforementioned costs of transmission time and computation time, we consider the security implications of task offloading for TO_i. Due to limited battery capacity and physical vulnerability of UAVs, concentrating tasks on a few resource-rich UAVs can negatively impact the system's performance and availability in case of a failure or attack on any one of those UAVs. Therefore, to enhance system robustness and mitigate the risk of single points of failure, we introduce a load balancing metric p_j, which increases with the number of tasks offloaded to resource UAV RP_j, representing the increasing probability of encountering a single point of failure. It can be defined as follows:

$$p_j = p_j^{ini} + p_j^{adj}(\sum_{i=1}^{I} \pi_i x_{ij}), \tag{18}$$

where, the load balancing metric p_j^{ini} is inversely proportional to the current battery level of RP_j. Considering that UAVs with lower battery levels have limited endurance, an excessive task burden can deplete their energy, affecting the reliability of task execution. Therefore, tasks are prioritized to be assigned to UAVs with higher battery levels to ensure stable system operation. On the other hand, p_j^{adj} is related to the inherent characteristics of the resource provider UAV itself. If the UAV possesses unique capabilities or performs critical tasks, this value will be higher. Due to variations in functionality or carried sensors among UAVs, some UAVs may be better suited for executing specific types of tasks in certain environments. These UAVs possess more valuable energy and computing resources, making their failure or malfunction pose a greater risk of system unavailability.

In addition, we also take into consideration the security issue of malicious nodes conducting fake task attacks. By combining the trust mechanism discussed in Section 4, the risk coefficient Δ_{ij} of offloading TO_i to RP_j is represented as the normalized ratio of node trustworthiness between RP_j and TO_i. A larger Δ_{ij} indicates that the task is offloaded from a low-trust node to a high-trust node, resulting in a higher risk of fake task attacks. Therefore, the cost of offloading also increases. Hence, the final cost function of TO_i under the offloading mode can be expressed as:

$$Cost_{TO_i}^{edge} = \sum_{j=1}^{J} (\xi \Delta_{ij}(t_{ij}^{d2d} + t_{ij}^{com}) + \Delta_{ij} p_j \pi_i) x_{ij}. \tag{19}$$

5.1.3. Total cost function

In practices the execution time of tasks which need to be offloaded is much longer than the consensus time, thus we neglect the cost of consensus time in our scenario.

By combining the task cost functions of TO_i under the two computing modes mentioned above, we derive the total cost function it imposes on the $AerialBC$, which can be represented by the following equation:

$$Cost_{TO_i} = Cost_{TO_i}^{edge} + \xi t_i^{local} x_i^{local}. \tag{20}$$

During the actual offloading decision-making process, TO_i minimizes task costs by adjusting its offloading decision $\boldsymbol{x_i} = [x_{i1}, x_{i2}, \dots, x_{iJ}, x_i^{local}]$ based on the conditions of offloading participants. This problem can be formulated as follows:

$$\begin{aligned} &\min_{\boldsymbol{x_i}} Cost_{TO_i}(\boldsymbol{x_i}) \\ s.t.\ &\begin{cases} \sum_{j=1}^{J} x_{ij} + x_i^{local} = 1, \\ x_{ij} \in \{0,1\},\ \forall j \in \{1,2,\dots,J\}, \\ x_i^{local} \in \{0,1\}. \end{cases} \end{aligned} \tag{21}$$

5.2. Task allocation strategy

The task allocation strategy aims to find the optimal task offloading decision for UAVs, in order to improve the resource utilization and robustness of UANET. Let $\boldsymbol{x_i}$ represent the offloading decision of TO_i, $\hat{\boldsymbol{x}}_i$ denote the offloading decisions of other task owner UAVs excluding TO_i, and the load balancing indicator p_j of RP_j is also determined by the decisions of other task owner UAVs, and we utilize $\boldsymbol{p}(\boldsymbol{x_i}, \hat{\boldsymbol{x}}_i)$ as the load balancing decision function for resource provider UAVs, which are influenced by the decision-making of the task owner UAVs. Therefore, for an individual TO_i's offloading decision problem, the optimization problem to be solved is as follows:

$$\begin{aligned} &\min_{\boldsymbol{x_i}} Cost_{TO_i}(\boldsymbol{x_i}, \hat{\boldsymbol{x}}_i, \boldsymbol{p}(\boldsymbol{x_i}, \hat{\boldsymbol{x}}_i)) \\ s.t.\ &\begin{cases} \sum_{j=1}^{J} x_{ij} + x_i^{local} = 1, \\ x_{ij} \in \{0,1\},\ \forall j \in \{1,2,\dots,J\}, \\ x_i^{local} \in \{0,1\}. \end{cases} \end{aligned} \tag{22}$$

For a TO_i, assuming the decisions of other task owner UAVs are fixed, the optimization objective of the TO_i is to minimize the task cost by adjusting its own decision. The computational power and risk cost allocated to the target resource provider UAV are affected by the decisions of other task owner UAVs. For example, when there are more tasks unloaded onto the same resource provider UAV, the lower the allocated computational power and the higher the risk. Therefore, the total completion cost increases. Conversely, when there are fewer tasks, the completion cost decreases. As a result, the decisions among task owner UAVs can ultimately reach an equilibrium state in which no task owner UAV can unilaterally modify its decision to reduce the task cost. Thus, the equilibrium state of this problem can be defined as the existence of an optimal decision $\boldsymbol{x_i^*}$ for any TO_i that satisfies the following inequality:

$$Cost_{TO_i}(\boldsymbol{p}(\boldsymbol{x_i^*}, \hat{\boldsymbol{x}}_i^*), \boldsymbol{x_i^*}, \hat{\boldsymbol{x}}_i^*) \geq Cost_{TO_i}(\boldsymbol{p}(\boldsymbol{x_i}, \hat{\boldsymbol{x}}_i^*), \boldsymbol{x_i}, \hat{\boldsymbol{x}}_i^*). \tag{23}$$

To address this issue, we design a task cost fast decline (TCFD) algorithm, as shown in Algorithm 1, to obtain optimal pricing and allocation strategies in equilibrium state, to meet the rapid decision-making requirements of the main nodes in time-sensitive UANET. First, we initialize an initial solution $(\boldsymbol{p}^{(0)}, \boldsymbol{x}^{(0)})$, where $x_i^{local,(0)} = 1, \forall i \in \{1,2,\dots,I\}$, indicating that all tasks of the UAVs are initially scheduled for local computation. Next, we set a maximum iteration count T and an iteration counter $\tau = 0$. Perform a maximum of T iterations, where the solution in each iteration is denoted as $(\boldsymbol{p}^{(\tau)}, \boldsymbol{x}^{(\tau)})$. The next iteration's solution is initially set to $(\boldsymbol{p}^{(\tau+1)}, \boldsymbol{x}^{(\tau+1)}) = (\boldsymbol{p}^{(\tau)}, \boldsymbol{x}^{(\tau)})$. Set a marker value $\Lambda = -1$, then iterate through all task owner UAVs. For each TO_i, calculate the task cost $Cost_{TO_i}^{cur}$ based on the decision in $(\boldsymbol{p}^{(\tau)}, \boldsymbol{x}^{(\tau)})$, and compute the minimum task cost $Cost_{TO_i}^{best}$ using the decision in $(\boldsymbol{p}^{(\tau)}, \hat{\boldsymbol{x}}_i^{(\tau)})$. If the difference between these two costs is greater than Λ, save this difference and record $\boldsymbol{x_i}^{(\tau+1)}$. After iterating through all task owner UAVs, update $(\boldsymbol{p}^{(\tau+1)}, \boldsymbol{x}^{(\tau+1)})$ with the last recorded $\boldsymbol{x_i}^{(\tau+1)}$. Then check if $(\boldsymbol{p}^{(\tau+1)}, \boldsymbol{x}^{(\tau+1)})$ is equal to $(\boldsymbol{p}^{(\tau)}, \boldsymbol{x}^{(\tau)})$. If they are equal, it indicates convergence of the decision; otherwise, continue to the next iteration until the maximum iteration count is reached.

5.3. Two-stage collaborative computing

Due to varying computational demands and resource availability of UAVs over time, the identities of task owner and resource provider UAVs are fixed for a period but differ across intervals. To accommodate this dynamism, the timeline is discretized into cycles of length τ. The master node periodically initiates collaboration at fixed intervals. For instance, during the ith cycle between times t_i and t_{i+1}, allocation decisions are made. After t_{i+1}, participants execute task offloading in a distributed manner. This allows nodes to flexibly become resource/task owners across intervals.

We divide collaboration into two stages: centralized "task allocation" and distributed "task offloading". Allocation must first be addressed to optimize performance and security. In distributed decision-making, stable states are reached through multi-round negotiations, which are less efficient and may fail to converge. Centralized decision-making enables quicker plans, improving efficiency. Thus, we adopt a

Algorithm 1: Task Cost Fast Decline algorithm

Initialize $(\boldsymbol{p}^{(0)}, \boldsymbol{x}^{(0)})$ with $x_i^{local,(0)} = 1, \forall i \in [1, I]$;
Set maximum iteration round T and current round $\tau = 0$;
while $\tau < T$ **do**
  $(\boldsymbol{p}^{(\tau+1)}, \boldsymbol{x}^{(\tau+1)}) \leftarrow (\boldsymbol{p}^{(\tau)}, \boldsymbol{x}^{(\tau)})$, $\Lambda \leftarrow -1$, $\Pi \leftarrow 0$;
  for $i \leftarrow 1$ **to** I **do**
    Calculate $Cost_{TO_i}$ for TO_i with $(\boldsymbol{p}^{(\tau)}, \boldsymbol{x}^{(\tau)})$ as $Cost_{TO_i}^{cur}$;
    Calculate optimal $Cost_{TO_i}$ for TO_i with $(\boldsymbol{p}^{(\tau)}, \hat{\boldsymbol{x}}_i^{(\tau)})$ as $(Cost_{TO_i}^{best}, \boldsymbol{x}_i^{(\tau+1)})$;
    if $(Cost_{TO_i}^{cur} - Cost_{TO_i}^{best}) > \Lambda$ **then**
      $\Lambda \leftarrow Cost_{TO_i}^{cur} - Cost_{TO_i}^{best}$, $\Pi \leftarrow \boldsymbol{x}_i^{(\tau+1)}$;
    end
  end
  Update $(\boldsymbol{p}^{(\tau+1)}, \boldsymbol{x}^{(\tau+1)})$ with Π;
  if $\left\|\boldsymbol{x}^{(\tau+1)} - \boldsymbol{x}^{(\tau)}\right\| == 0$ **then**
    break;
  end
  $\tau \leftarrow \tau + 1$;
end

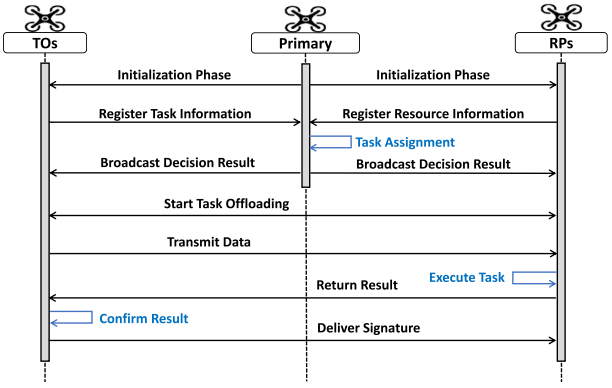


Fig. 3. Two-stage collaborative computing process in UANET.

centralized approach with a highly trusted master node for decision-making to mitigate single point failures. It collects information, receives participant data, and executes the algorithm. Subsequently, participants proceed with offloading by invoking smart contracts, ensuring standardized execution. Deviations by malicious nodes are detected and recorded during offloading, decreasing their trust. Nodes are expelled once negative thresholds are reached. Fig. 3 shows the two-stage collaboration. Details are:

(1) Task Allocation. The collaborative computing cycle begins as the consensus master node establishes a new task allocation contract. Nodes willing to participate in collaborative computing, including those with task demands and surplus computational power, share their collaborative information with the master node through this smart contract. Prior to the completion of the task allocation process, the master node executes the task allocation algorithm and uploads the decision outcome to the contract for dissemination to all participating nodes. Based on the master node's decision, each participating node proceeds with the subsequent task collaboration process.

(2) Task Offloading. The task offloading process is initiated as the task owner UAV and resource provider UAV jointly create a task offloading contract through mutual agreement. Subsequently, following the contract rules, both parties engage in distributed processes such as data transmission, task computation, and result feedback. Upon receiving the computation results, the task owner UAV verifies them and sends an acknowledgment signature to the resource provider UAV. The details or summaries of the computational tasks, results, and acknowledgment signatures are uploaded and queried through the smart contract to ensure traceability in the collaborative computing process, providing a reference for trust evaluation in Section 4.

5.4. Smart contract design

Based on the description of aforementioned collaborative computing process, design corresponding smart contracts. There are primarily three types of contracts: task allocation contract, task offloading contract, and accusation contract. The task allocation contract is created by the master node at the beginning of each cycle. The task offloading contract is jointly signed by the task owner UAV and resource provider UAV before the task offloading process commences. The accusation contract is created by the accusing UAV that suspects the behavior of other UAVs.

5.4.1. Task allocation contract

The task allocation contract is primarily used for information gathering and task allocation decisions by the master node. In this process, task owner UAVs register their task attributes using the *taskOwnerRegister* function, while resource provider UAVs register their computing resources and load balancing metrics using the *resourceProviderRegister* function. After collecting network information and coordinating with participating UAVs in the computation, the master node executes the task allocation algorithm and uploads all the information, along with the task allocation results, using the *setPrimaryResult* function. The various participants in collaboration can view the task allocation results through the *getPrimaryResult* function, as shown in Algorithm 2.

Algorithm 2: Task Allocation Contract

create():
  The master node creates a task allocation contract for the current cycle;
taskOwnerRegister$(s_i, \kappa_i, t_i^{\max}, f_i^{local})$:
  Task owner UAV provides the registration information and submits $\{s_i, \kappa_i, t_i^{\max}, f_i^{local}\}$;
resourceProviderRegister$(f_j^{cap}, p_j^{ini}, p_j^{adj})$:
  Resource provider UAV register and submits $\{f_j^{cap}, p_j^{ini}, p_j^{adj}\}$;
setPrimaryResult$(result)$:
  The primary node uploaded the task allocation results;
getPrimaryResult():
  The participants in the collaborative computing get the task allocation results;

5.4.2. Task offloading contract

Upon mutual confirmation by the task offloading parties, a task offloading contract is jointly signed by the task owner UAV and the resource provider UAV. Subsequently, the task digest is uploaded via the *submitTaskDigest* function, while the actual task data can be transmitted offline to the resource provider UAV. Once the computation is completed, the result digest is uploaded through the *submitResultDigest* function and returned offline as well. The task owner UAV verifies the results, uploads the reception signature using the *submitResultSign* function, and concludes the task offloading process, as shown in Algorithm 3.

5.4.3. Accusation contract

When UAV_i suspects the behaviors of UAV_j (*e.g.*, failure to unload tasks as per assignment), UAV_i initiates a suspicion transaction by creating an accusation contract. Other UAVs vote on the accusation proposal through *Vote* function, and the contract internally invokes *CountVote* function to tally the voting results based on the

Algorithm 3: Task Offloading Contract

create($PK_i, PK_j, time, location, Evidence, Sig_i$):
    Task owner UAV and resource provider UAV jointly sign to create a task offloading contract;
submitTaskDigest($taskDigest$):
    The task owner UAV submits a summary of the task;
submitResultDigest($resultDigest$):
    The resource provider UAV submits a summary of the results;
submitResultSign($sign$):
    The task owner UAV submits an acknowledgment signature, concluding the task offloading process;

proposal scheme in Section 4. If the vote count passes, the master node broadcasts the accused UAV's misconduct along with the signatures of the voting UAVs via block propagation through *RecordMisbehaviour* function, as shown in Algorithm 4.

The primary function of smart contracts revolves around meticulously recording allocation inputs and outputs, allowing us to largely disregard computational and communication loads considerations when conducting evaluation.

Algorithm 4: Accusation Contract

create(Sig_i,Sig_j):
    The accusing UAV initiates a suspicion transaction;
Vote(Sig):
    Other UAVs participate in the voting process;
CountVote():
    The contract internally employs a proposal voting scheme to calculate the voting results;
RecordMisbehaviour($Sigs$):
    Once the voting results are approved, the master node records the misconduct in the blockchain;

6. Performance evaluation

6.1. Security analysis

6.1.1. Security analysis framework

In order to provide a structured and comprehensive security analysis of the *AerialBC* framework, we adopt a multi-faceted security analysis framework that encompasses the following components:

Threat Modeling: Identifying potential threats and vulnerabilities within the UANET environment, such as single point failures, false task attacks, and active malicious nodes.

Risk Assessment: Evaluating the likelihood and impact of identified threats materializing, and determining the risk levels associated with different network operations.

Security Mechanisms: Detailing the specific security mechanisms employed to counter identified threats, including cryptographic methods, consensus algorithms, trust evaluation, and network protocols.

Validation and Verification: Conducting simulations and theoretical analyses to validate the effectiveness of security mechanisms, and verifying their performance against industry-standard security benchmarks.

Incident Response: Outlining the procedures for detecting, responding to, and recovering from security incidents, ensuring the UANET's resilience and continuity of operations.

Using this framework, we systematically address the security concerns within the UANET by implementing robust security mechanisms and protocols, and provide a thorough analysis of their effectiveness.

6.1.2. Security analysis results

Through a combination of attack simulations, we demonstrate the system's ability to withstand single points of failure, mitigate false task attacks, and prevent the actions of active malicious nodes.

(a) Preventing Single Point Failure. By introducing the load balancing metric p, we can assess the workload of UAVs in UANET. When tasks are concentrated on a few UAVs, the value of the load balancing metric increases, indicating an imbalanced task allocation and a higher risk of system unavailability due to the failure or malfunction of a single UAV. In task allocation strategy, we consider the load balancing metric an essential parameter for measuring task cost. By minimizing task costs, we balance task execution efficiency and task offloading safety, effectively reducing the risk of single-point failures in UANET. As for the blockchain scheme, due to the inherently distributed nature of the blockchain itself and the committee period set in the consensus algorithm proposed in this paper, UANET nodes in the blockchain network can dynamically join and leave, effectively coping with the dynamic changes in the blockchain node caused by single point failures.

(b) Mitigating False Task Attack. UAVs with lower trustworthiness, assessed by a trust evaluation mechanism, are more prone to inappropriate behavior, including non-cooperation and false message transmission. These UAVs are categorized as potential malicious nodes. To mitigate the influence of false task attacks initiated by such nodes on system security and performance, we introduce a risk coefficient Δ_{ij} in the task cost computation. This implies that, through task allocation strategies, UAVs with lower trustworthiness encounter challenges when offloading tasks to UAVs with higher trustworthiness. Instead, they tend to collaborate with UAVs of similar trustworthiness, reducing the risk of resource wastage among highly trustworthy nodes. UAVs accumulating negative behavior due to non-compliance with task allocation rules face expulsion from the UANET if their trustworthiness drops below a specified threshold. These methods help prevent malicious nodes from gradually gaining control over the network.

(c) Preventing Active Malicious Nodes. UAV nodes' trustworthiness in the consensus and network is closely linked to their level of participation and activity. For example, absentee voting on proposals is considered negative behavior. Hence, active engagement by normal UAV nodes is essential to reduce the impact of actively malicious nodes.

6.2. Lightweight blockchain verification

6.2.1. Simulation setup

The tests involved simulating a UANET on a 12 vCPU, 24GiB memory server running Ubuntu 18.04 with Intel Cascade Lake 2.6 GHz CPUs. To simulate UAVs within the UANET, different ports were internally used on the server. A modified Python implementation of PBFT enabled the consensus process. Multiple miner and observer nodes were launched. UANET sizes were set to 4, 12, 20, 28, 36, 44, 52 and 60, with a maximum 20 consensus nodes. When UAVs were less than 20, all participated in consensus. For over 20 UAVs, only the top 20 nodes by trust value were selected as consensus nodes, since consensus complexity relates to committee size. Each node submitted collaboration requests at a fixed frequency during consensus. A standard PBFT consensus was used for comparison, primarily evaluating consensus latency and message count to highlight the performance advantages of our proposed approach.The experiments focused on how to improve efficiency and reduce overhead for the lightweight blockchain consensus, rather than overall UANET performance.

6.2.2. Results and analysis

Fig. 4 illustrates the variation of trust values for a single malicious node across consensus rounds, showcasing the operation of the designed mechanism for evaluating node trustworthiness. The node engaged in three negative or malicious behaviors out of a total of 1000 consensus rounds. Despite accumulating a considerable amount of trust through actions such as participating in proposal voting, the

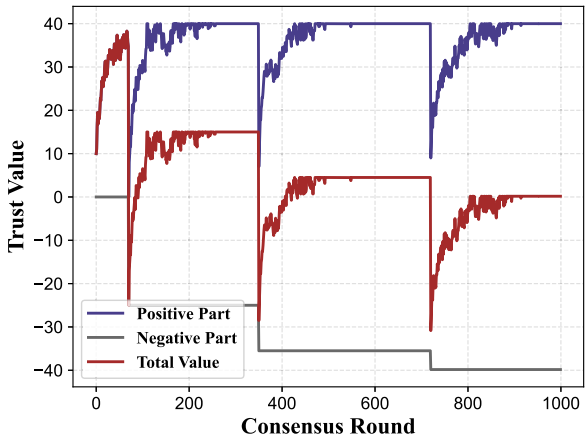


Fig. 4. Variation of trust values for a malicious UAV across consensus rounds.

upper bound of its trustworthiness decreases as its negative behavior increases. This effectively assesses the trustworthiness of malicious or unstable nodes, enabling rapid differentiation of trust levels among UANET nodes.

Fig. 5 displays the proportion of malicious miners over consensus rounds, comparing the proposed *Aerial BC* consensus to PBFT and PoW. Subfigures (a)–(d) show initial malicious node proportions of 0.01, 0.1, 0.2 and 0.3, evaluating malicious miner proportions every 100 rounds. For PoW and PBFT, the malicious miner proportion roughly equals the initial malicious nodes due to miner selection based on computation and primary node rotation. In contrast, *Aerial BC*'s trust evaluation mechanism enables quick identification of malicious nodes, significantly reducing malicious miners and enhancing security. Nodes with more negative behaviors have lower trust, making miner committee entry difficult, effectively safeguarding consensus. Thus, the trust mechanism allows *Aerial BC* to adaptively adjust malicious miner proportions. Fig. 6(a) shows consensus latency versus UANET size, comparing the PBFT-based method to the proposed *Aerial BC*. Results show our method significantly reduces transaction propagation latency by reducing consensus nodes. With 20 committee nodes, both methods have similar latency for under 20 UAVs. However, above 20 UAVs, the reduced committee yields substantially decreased latency, averaging 47% lower across scales. This is not just interactions among the 20 UAVs, but communication with the cluster, implying cluster consensus adapts accordingly as nodes increase. In essence, limiting the committee enables *Aerial BC* to maintain consensus efficiency even as the overall UANET grows, overcoming bottlenecks facing standard PBFT implementations. Fig. 6(b) exhibits consensus message count versus UANET size for the PBFT method and *Aerial BC*. Results demonstrate *Aerial BC* effectively reduces messages. Analogous to latency, when UAVs exceed 20, there is a major decrease in messages, averaging 76% lower across scales. By optimizing latency and messages, *Aerial BC* better adapts to UANET's constraints on computation and scalability compared to standard PBFT. Overall, our *Aerial BC*, demonstrates significant improvements in efficiency and security for collaborative computing in UANET. Compared to [13,14], and [15], *Aerial BC* fills a critical gap by providing a comprehensive, lightweight, and secure blockchain framework tailored for UANET, ensuring efficient resource utilization and robust security in highly dynamic environments.

6.3. Task allocation strategy verification

6.3.1. Simulation setup

We randomly generate UAV nodes in 3D space, with 5–40 task owner UAVs and 6–24 resource provider UAVs. Connectivity is checked based on node communication ranges. If disconnected components exist, new nodes are added to connect all nodes into a single UANET. A minimum safe inter-UAV flight distance is maintained. For simulations, we assume link quality between nodes i and j depends on their relative distance and velocity, setting it as:

$$ETX_{ij} = \begin{cases} e^{v_{ij}\gamma}, & 0 \le d_{ij} \le d_c/2 \\ e^{(2d_{ij}/d_c)-1} \cdot e^{v_{ij}\gamma}, & d_c/2 \le d_{ij} \le d_c \\ \infty, & d_{ij} > d_c \end{cases} \tag{24}$$

where, d_{ij} and v_{ij} denote the relative distance and relative velocity between two UAVs respectively, with γ set to 0.3.

In the UANET, certain UAVs are selected as task owners or resource providers for collaborative computing, with remaining UAVs participating as routing relay nodes. Figs. 7(a) and 7(b) illustrate the generated 3D and 2D top-down UANET topology and link quality, respectively. Red triangles are resource providers, green crosses are task owners, and black points are relays. Dashed lines show wireless links between UAVs, with darkness indicating quality. Once the topology and links are obtained, shortest routes are calculated via Floyd's algorithm. Thereafter, task allocation decisions are made based on collaboration information between involved UAVs. Overall, the simulations aim to evaluate collaborative computing performance given realistic wireless environments. Key parameter values used in experiments are presented in Table 2. Allocated capacities for idle UAVs are 3–10 times the local capacity of a single task owner. Other parameters are randomly set within defined ranges.

Table 2
The values of key parameters.

Symbol	Value [unit]
s_i	$1.2 * 10^7$–$1.5 * 10^7$ [bit]
κ_i	10000 [cycle/bit]
$t_i^{\max}$	180–220 [s]
ξ	0.1–0.3
v_η	0–2 [m/s]
d_c	30 [m]
f_i^{local}	$2 * 10^8$–$5 * 10^8$ [Hz]
Δ_{ij}	0.9–1.1
ϵ_j	$2.5 * 10^{-8}$
p_j^{ini}	$6 * 10^{-9}$
p_j^{adj}	$6 * 10^{-18}$
f_j^{cap}	$1.5 * 10^9$–$2 * 10^9$ [Hz]

6.3.2. Results and analysis

(a) Task Offloading Results. Figs. 8(a) and 8(b) depict an example task offloading assignment from 3D and 2D top-down views. Red nodes are resource providers, green nodes are task owners, with solid black lines showing allocation. There are 20 task owners and 12 resource providers. Most offloading requirements are fulfilled by the resources. Spatially, tasks tend to be offloaded to nearby providers. However, if nearby providers are heavily loaded amid intense computational competition, distant idle providers may be chosen instead. This demonstrates the strategy's capability of globally optimizing allocation decisions across the UANET topology based on individual provider loads and link quality between nodes.

(b) Total Task Cost under different Allocation Strategies. Fig. 9(a) compares total task costs under different allocation strategies, with up to 24 resource provider UAVs (RPs) and 40 task owner UAVs (TOs). Costs include UANET operation and risk, comparing: Local Computing (LC), Random Offloading (RD), Nearest Neighbor by Position (NN_Pos), Nearest Neighbor by ETX (NN_Etx), and our proposed algorithm (TCFD). Under LC, TOs perform local computing, hence cost does not depend on RPs. Costs for NN_Pos and NN_Etx vary significantly with RPs. With fewer RPs, neighbor allocation has higher costs due to overloading nearest RPs. This improves with more RPs. RD

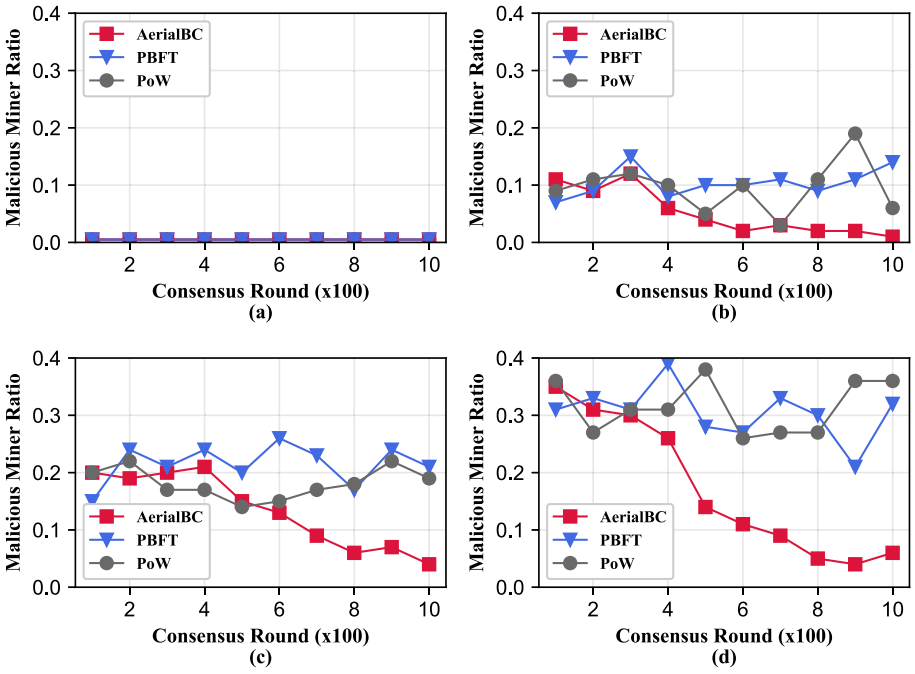


Fig. 5. Proportion of malicious miners over consensus rounds.

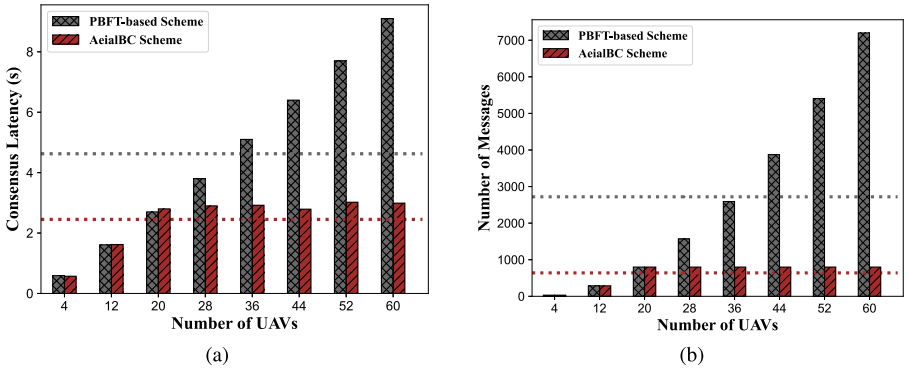


Fig. 6. Consensus latency and message count against the variation of UANET size.

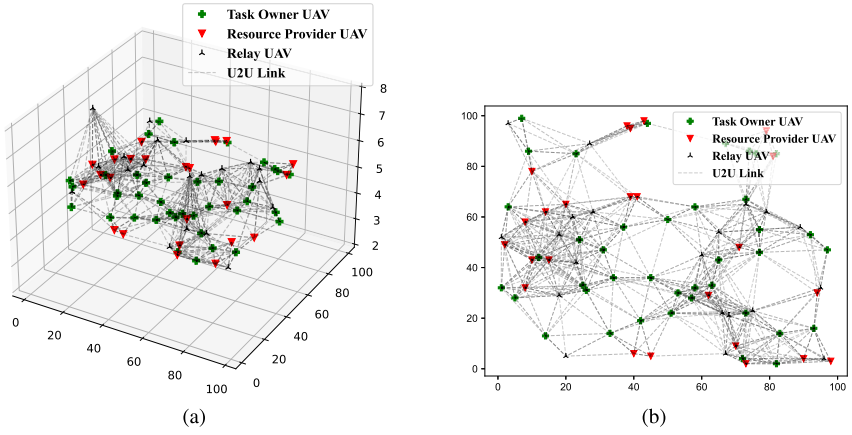


Fig. 7. 3D and 2D view of the generated UANET topology and link. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

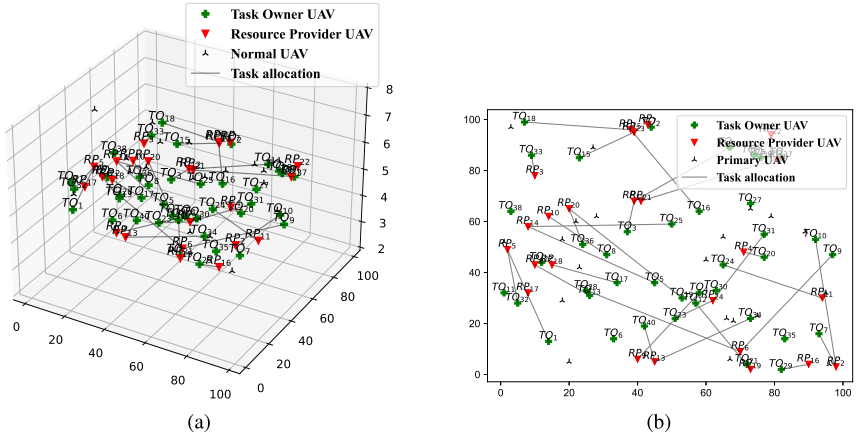


Fig. 8. 3D and 2D view of the task offloading results. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

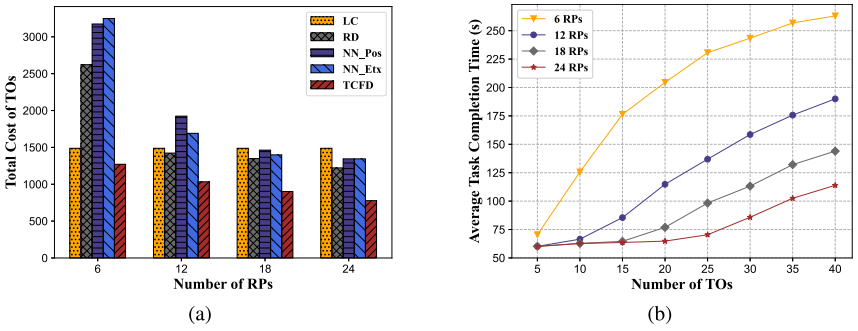


Fig. 9. (a) Comparison of task costs under different allocation strategies, (b) Average task completion time of TOs in UANET with varying numbers of RPs.

among providers is especially effective for more RPs, lowering costs. TCFD consistently performs well across RP quantities. The spatial optimization and dynamic load balancing enable efficient global allocation. In a 24 RP UANET, TCFD reduces TO costs by 48% over LC.

(c) Average Task Completion Time. Fig. 9(b) shows the average task completion time of TOs versus RPs in the UANET. Completion time represents overall TO efficiency. With the same RPs, more TOs increase computational resource demand, intensifying competition. Individual RPs bear more offloaded tasks, dispersing power across tasks and raising risks, prompting TOs to favor local computing. This increases costs and completion times. In contrast, with the same TOs, more RPs augment total computational resources, encouraging offloading mode usage, efficiently utilizing provider capacities and reducing duration. For instance, with 40 TOs and 6 RPs, average time is 217 s. However, 24 RPs drops this to 99 s, a 57% reduction. Essentially, while TO scalability heightens resource contention, RP scalability alleviates it. TCFD optimally balances load across ample global resources.

(d) Time Cost and Offloading Rate. Fig. 10 shows the task offloading rate versus different task time cost coefficients ξ, representing costs per unit execution time. Higher coefficients imply stronger willingness for owners to offload and reduce time, while lower coefficients suggest the opposite. With the same RPs, larger coefficients lead to higher overall offloading. However, with low coefficients, even more RPs barely increase offloading, because the strategy favors lower local computing time costs over offloading costs, encouraging local execution. Essentially, the time coefficient allows adjusting the collaborative computing degree. When minimizing execution time is critical, increasing the coefficient promotes offloading to utilize ample global resources for faster parallel execution. But for latency-tolerant applications, local computing may be preferred, sacrificing time for efficiency. Overall, the time coefficient provides a tuning knob to achieve the desired tradeoff based on application requirements. This enables configurability and customization within the UANET.

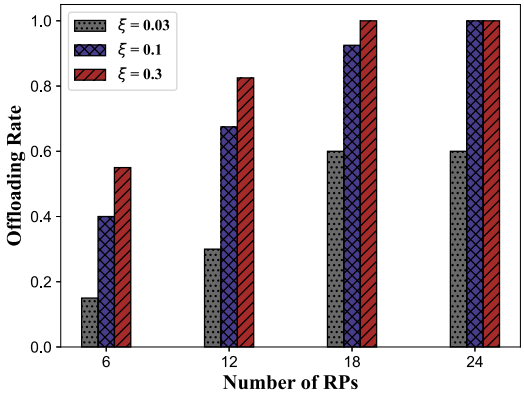


Fig. 10. Impact of different task time cost coefficients ξ on the task offloading rate in UANET.

7. Practical evaluation

In this section, we implement *Aerial BC* in a real scene and evaluate its performance.

7.1. Experiment settings

As shown in Fig. 11, the system hardware modules mainly consist of a swarm of 4 small rotary-wing UAVs, each equipped with a Raspberry Pi board[1] as the onboard computing and communication module. The Raspberry Pi is connected to the UAV's Pixhawk flight controller via

[1] Raspberry Pi: https://www.raspberrypi.org/.

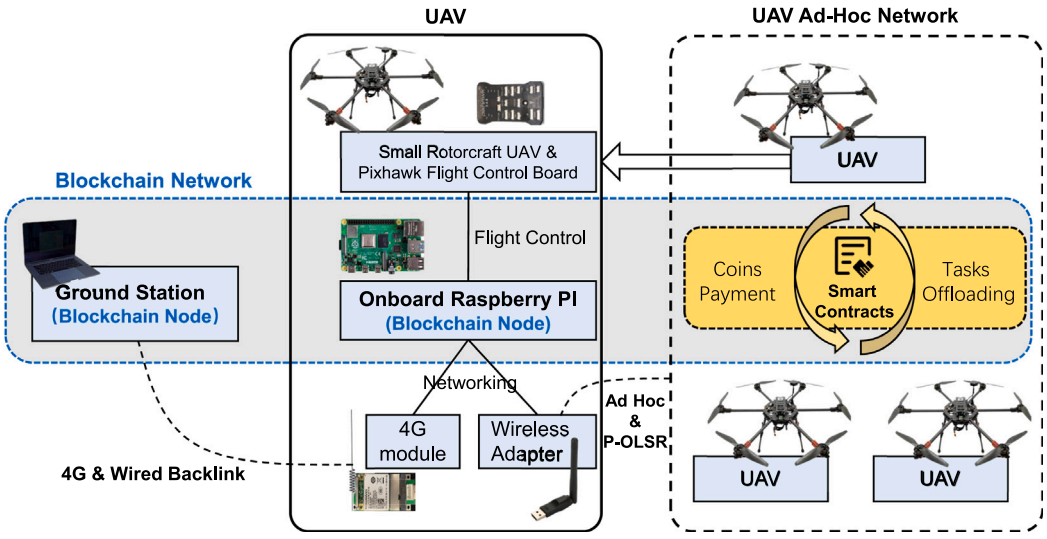


Fig. 11. Framework of *AerialBC* real-world system.

a serial port and controls the UAV flight using the Dronekit SDK.[2] The Raspberry Pi is equipped with a Ralink RT5730 wireless adapter that supports Ad hoc mode and is connected to the Raspberry Pi via a USB port. When the wireless adapter is set to Ad hoc mode, it can be used to form a self-organizing network within the UAV swarm, and then routing protocols are run to provide routing information and link quality information to the nodes.

The blockchain network is built on the basis of UANET, with the Raspberry Pis running Ethereum clients to form a private blockchain network through node discovery. The main workflows such as computing power trading and collaborative computing also run on the Raspberry Pi. In addition, a HUAWEI ME909S-821 Mini PCIe 4G module and antenna are connected to the Raspberry Pi, which connects to a 4G base station via wvdial to communicate with the ground station. A laptop is used as the ground station, responsible for controlling the swarm flight, monitoring the blockchain network, and issuing resource tokens. After the UAVs take off and run, the UANET can be considered to enter an autonomous state, and the ground station only acts as a monitoring node, with its connection to the swarm regarded as a weak connection. Additionally, considering that the UAVs and ground station are in their respective intranet environments during the actual networking process, an additional server with a public IP is required to act as a network relay.

Table 3
Main node creates pricing allocation contract.

transaction hash	0xbfc330e5330ba3c63dc17e99f1b7f8fdf39d001679005ac0af263bc34514e300
from	0xA4F2f09833778518dC6422C69a87fd17970158a0
to	PricingAllocationContract.create 0x9bF88fAe8CF8BaB76041c1db6467E7b37b977dD7
input	{}
output	{}

Table 4
Task node registers information.

transaction hash	0xf67d673d367694bfb284bb51289e0a55f09b1a1d3f9582d8d7d371b27b44695e
from	0x2A3803A4a03bE2363e632323dC78e18df1467d45
to	PricingAllocationContract.taskOwnerRegister 0x9bF88fAe8CF8BaB76041c1db6467E7b37b977dD7
input	{"task_s":5,"task_k":8700,"task_t_max":70,"f_local":15,"cost_coefficient":15}
output	{}

7.2. Real-world testing and result analysis

Based on the above hardware and software modules, a prototype system of AerialBC is implemented, and real-world testing is conducted in this section. In terms of the test workload, the classic Traveling Salesman Problem (TSP) is used as a task instance, which often appears in applications such as UAV point-of-interest path planning. Using the dynamic programming method as the baseline algorithm, two task loads, TSP-20n-10x and TSP-20n-5x, are constructed, representing continuous computation of 10 and 5 TSP problems containing 20 path points, respectively. Assuming that the task UAV's local computing power is 10% remaining, tests show that using 10% of the task UAV's Raspberry Pi CPU computing power to complete TSP-20n-10x takes about 290 s, and TSP-20n-5x takes about 145 s, based on which the task parameters are set.

Part of the contract invocation transaction information in the transaction and computing process are given in table form to show the actual contract invocation process. The main node creates a pricing allocation contract to open the transaction for the current period, as shown in Table 3. The TO node invokes the `taskOwnerRegister` function of the pricing allocation contract to register task information, as shown in Table 4. Both RP_1 and RP_2 nodes invoke the `resourceProviderRegister` function of the pricing allocation contract to register computing power and pricing information, as shown in Tables 5 and 6. The staking and payment of resource tokens during the transaction process are shown in Tables 7 and 8.

The statistical results of the actual test are shown in Table 9. For the task load TSP-20n-5x, where the local mode is expected to take 145 s, TO_1 reduces the actual task time to 21.85 s by purchasing RP_1's computing power and performing computational offloading, while paying 12.21 units of resource tokens. For the task load TSP-20n-10x, where the local mode is expected to take 290 s, TO_1 reduces the actual task time to 39.33 s by purchasing RP_1's computing power and performing computational offloading, while paying 30.22 units of resource tokens.

As shown in Fig. 12, the comparison of the time cost of the TO node in local mode and the time cost and payment cost in offload mode during the actual test. It can be seen that by paying a certain

[2] Dronekit SDK: https://dronekit.io/.

Table 5
RP_1 registers information.

transaction hash	0x56c6b7534abe3c9e59caecad8dd0f96bc3cdddb481e035f77f9fbff264ed8b31
from	0xCCd85C96D370B3D1beF1d069a560c0110EC2a577
to	PricingAllocationContract.resourceProviderRegister 0x9bF88fAe8CF8BaB76041c1db6467E7b37b977dD7
input	{"f_cap":120,"cost_j":1,"p_ini":3,"p_adj":3}
output	{}

Table 6
RP_2 registers information.

transaction hash	0x3648e18b43407701882bfd126ce8e8bd3bf41b9f93e2fee7967e69329f327287
from	0x4d54faaB2E1A0910D3b1f8B18aC95731e653CC52
to	PricingAllocationContract.resourceProviderRegister 0x9bF88fAe8CF8BaB76041c1db6467E7b37b977dD7
input	{"f_cap":120,"cost_j":1,"p_ini":3,"p_adj":3}
output	{}

Table 7
Token stake information.

transaction hash	0xf0d491c6814f8e49f15a019ae0a6636151e2ff31b4daa745d634ff95541a517a
from	0x2A3803A4a03bE2363e632323dC78e18df1467d45
to	ResourceCoinContract.deposit 0xf8e81D47203A594245E36C48e151709F0C19fBe8
input	{"_primary":"0xA4F2f09833778518dC6422C69a87fd17970158a0","_value":"3022"}
output	{}

Table 8
Token delivery information.

transaction hash	0xd5ec1010c13fb0a0adf8ee750ead41584c1c647a446f29aa0bae8e742d5d1a56
from	0xA4F2f09833778518dC6422C69a87fd17970158a0
to	ResourceCoinContract.transferDeposit 0xf8e81D47203A594245E36C48e151709F0C19fBe8
input	{"_to":"0xCCd85C96D370B3D1beF1d069a560c0110EC2a577","_value":"3022"}
output	{}

Table 9
Actual test results.

Task workload	TSP-20n-5x	TSP-20n-10x
Estimated time in local mode	≈145 [s]	≈290 [s]
Actual time in offloading mode	≈21.85 [s]	≈39.33 [s]
Transaction partner	RP_1	RP_1
Payment cost	12.21 [coins]	30.22 [coins]

amount of resource tokens, the TO node can offload the task to the RP node when its own remaining computing power is insufficient, greatly reducing the time cost for task completion and improving task efficiency, with an overall cost lower than the local computing cost. For the RP node, it obtains resource token incentives by providing computing power resources, which serves as proof of its participation in system computing. The tokens are recorded in the distributed ledger of the UAV blockchain network and can be used to purchase computing power when the RP node has task demands, effectively balancing the internal computing power resources and demand distribution of the UANET and limiting the impact of malicious and selfish nodes on the system.

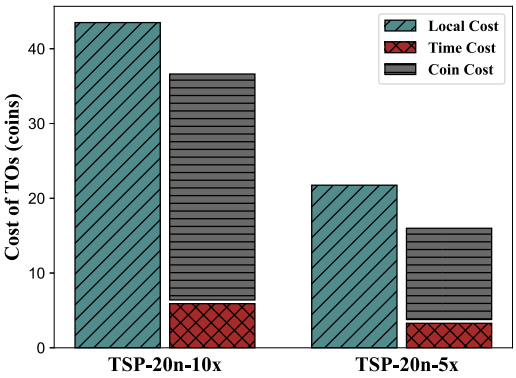


Fig. 12. Comparison of TO node costs in actual tests.

8. Conclusion

In this paper, we propose *Aerial BC*, a lightweight blockchain framework for secure UANET collaborative computing. First, an improved PBFT consensus based on trust evaluation deploys a blockchain recording UAV behaviors and collaborations while reasonably assessing trustworthiness. Next, a task allocation strategy balancing efficiency and security is presented. Smart contracts enable a two-stage process where owners optimally decide and standardly offload tasks. Experiments demonstrate *Aerial BC* efficiently operates within UANET resource constraints while promoting secure, efficient UAV collaboration. Overall, *Aerial BC* provides a decentralized platform leveraging blockchain and smart contracts to facilitate configurable, trustworthy, and scalable coordinated aerial computing.

CRediT authorship contribution statement

Runqun Xiong: Writing – review & editing, Supervision, Resources, Funding acquisition, Formal analysis. **Qing Xiao:** Writing – original draft, Software, Investigation, Formal analysis. **Zhoujie Wang:** Software, Methodology, Data curation. **Zhuqing Xu:** Writing – review & editing, Validation, Project administration. **Feng Shan:** Writing – review & editing, Validation, Formal analysis.

Declaration of competing interest

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

Data availability

Data will be made available on request.

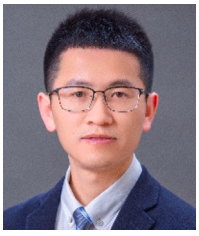
Acknowledgments

This work is supported by the National Natural Science Foundation of China under grants 62172091, 62232004, 61602112, and 61632008; the Jiangsu Provincial Key Laboratory of Network and Information Security, China under grant BM2003201; and the Key Laboratory of Computer Network and Information Integration of the Ministry of Education of China, China under grant 93K-9.

References

[1] Ivan Maza, Fernando Caballero, Jesus Capitan, J.R. Martinez-de Dios, Anibal Ollero, Experimental results in multi-UAV coordination for disaster management and civil security applications, J. Intell. Robot. Syst. 61 (1-4, SI) (2011) 563–585.
[2] Lav Gupta, Raj Jain, Gabor Vaszkun, Survey of important issues in UAV communication networks, IEEE Commun. Surv. Tutor. 18 (2) (2016) 1123–1152.
[3] Zhiqing Wei, Mingyue Zhu, Ning Zhang, Lin Wang, Yingying Zou, Zeyang Meng, Huici Wu, Zhiyong Feng, UAV-assisted data collection for internet of things: A survey, IEEE Internet Things J. 9 (17) (2022) 15460–15483.

[4] Alessio Rugo, Claudio A. Ardagna, Nabil El Ioini, A security review in the UAVNet era: Threats, countermeasures, and gap analysis, ACM Comput. Surv. 55 (1) (2022) 1–35.
[5] Yuntao Wang, Zhou Su, Qichao Xu, Ruidong Li, Tom H. Luan, Lifesaving with RescueChain: Energy-efficient and partition-tolerant blockchain based secure information sharing for UAV-aided disaster rescue, in: 2021 IEEE Conference on Computer Communications, INFOCOM, 2021, pp. 1–10.
[6] Rooha Masroor, Muhammad Naeem, Waleed Ejaz, Resource management in UAV-assisted wireless networks: An optimization perspective, Ad Hoc Netw. 121 (2021) 102596.
[7] Quyuan Luo, Tom H. Luan, Weisong Shi, Pingzhi Fan, Deep reinforcement learning based computation offloading and trajectory planning for multi-UAV cooperative target search, IEEE J. Sel. Areas Commun. 41 (2) (2023) 504–520.
[8] Kai-Yun Tsao, Thomas Girdler, Vassilios G. Vassilakis, A survey of cyber security threats and solutions for UAV communications and flying ad-hoc networks, Ad Hoc Netw. 133 (2022) 102894.
[9] Pavlos Athanasios Apostolopoulos, Georgios Fragkos, Eirini Eleni Tsiropoulou, Symeon Papavassiliou, Data offloading in UAV-assisted multi-access edge computing systems under resource uncertainty, IEEE Trans. Mob. Comput. 22 (1) (2023) 175–190.
[10] Haifeng Yu, Ivica Nikolić, Ruomu Hou, Prateek Saxena, OHIE: Blockchain scaling made simple, in: 2020 IEEE Symposium on Security and Privacy, SP, 2020, pp. 90–105.
[11] Yu Du, Zhe Wang, Jun Li, Long Shi, Dushantha Nalin K. Jayakody, Quan Chen, Wen Chen, Zhu Han, Blockchain-aided edge computing market: Smart contract and consensus mechanisms, IEEE Trans. Mob. Comput. 22 (6) (2023) 3193–3208.
[12] Wenyu Li, Chenglin Feng, Lei Zhang, Hao Xu, Bin Cao, Muhammad Ali Imran, A scalable multi-layer PBFT consensus for blockchain, IEEE Trans. Parallel Distrib. Syst. 32 (5) (2021) 1146–1160.
[13] Zhenzhen Guo, Gaoli Wang, Yingxin Li, Jianqiang Ni, Guoyan Zhang, Attribute-based data sharing scheme using blockchain for 6G-enabled VANETs, IEEE Trans. Mob. Comput. 23 (4) (2024) 3343–3360.
[14] Xin Xie, Cunqing Hua, Jianan Hong, Pengwenlong Gu, Wenchao Xu, AirCon: Over-the-air consensus for wireless blockchain networks, IEEE Trans. Mob. Comput. 23 (5) (2024) 4566–4582.
[15] Xiao Chen, Guoliang Xue, Ruozhou Yu, Haiqin Wu, Dawei Wang, A vehicular trust blockchain framework with scalable Byzantine consensus, IEEE Trans. Mob. Comput. 23 (5) (2024) 4440–4452.
[16] Zhou Su, Yuntao Wang, Qichao Xu, Ning Zhang, LVBS: Lightweight vehicular blockchain for secure data sharing in disaster rescue, IEEE Trans. Dependable Secure Comput. 19 (1) (2022) 19–32.
[17] Mohammed Seid Abegaz, Hayla Nahom Abishu, Yasin Habtamu Yacob, Tewodros Alemu Ayall, Aiman Erbad, Mohsen Guizani, Blockchain-based resource trading in multi-UAV-assisted industrial IoT networks: A multi-agent DRL approach, IEEE Trans. Netw. Serv. Manag. 20 (1) (2023) 166–181.
[18] Xiaobin Xu, Hui Zhao, Haipeng Yao, Shangguang Wang, A blockchain-enabled energy-efficient data collection system for UAV-assisted IoT, IEEE Internet Things J. 8 (4) (2021) 2431–2443.
[19] Rui Xing, Zhou Su, Tom Hao Luan, Qichao Xu, Yuntao Wang, Ruidong Li, UAVs-aided delay-tolerant blockchain secure offline transactions in post-disaster vehicular networks, IEEE Trans. Veh. Technol. 71 (11) (2022) 12030–12043.
[20] Shuyun Luo, Hang Li, Zhenyu Wen, Bin Qian, Graham Morgan, Antonella Longo, Omer Rana, Rajiv Ranjan, Blockchain-based task offloading in drone-aided mobile edge computing, IEEE Netw. 35 (1) (2021) 124–129.
[21] Abegaz Mohammed Seid, Jianfeng Lu, Hayla Nahom Abishu, Tewodros Alemu Ayall, Blockchain-enabled task offloading with energy harvesting in multi-UAV-assisted IoT networks: A multi-agent DRL approach, IEEE J. Sel. Areas Commun. 40 (12) (2022) 3517–3532.
[22] Xiao Tang, Xunqiang Lan, Lixin Li, Yan Zhang, Zhu Han, Incentivizing proof-of-stake blockchain for secured data collection in UAV-assisted IoT: A multi-agent reinforcement learning approach, IEEE J. Sel. Areas Commun. 40 (12) (2022) 3470–3484.
[23] Chunpeng Ge, Xinshu Ma, Zhe Liu, A semi-autonomous distributed blockchain-based framework for UAVs system, J. Syst. Archit. 107 (2020) 101728.
[24] Rajesh Gupta, Anuja Nair, Sudeep Tanwar, Neeraj Kumar, Blockchain-assisted secure UAV communication in 6G environment: Architecture, opportunities, and challenges, IET Commun. 15 (10) (2021) 1352–1367.
[25] Pramod Abichandani, Deepan Lobo, Smit Kabrawala, William McIntyre, Secure communication for multiquadrotor networks using Ethereum blockchain, IEEE Internet Things J. 8 (3) (2021) 1783–1796.
[26] Keke Gai, Yulu Wu, Liehuang Zhu, Kim-Kwang Raymond Choo, Bin Xiao, Blockchain-enabled trustworthy group communications in UAV networks, IEEE Trans. Intell. Transp. Syst. 22 (7) (2021) 4118–4130.
[27] Baichuan Liu, Weikun Zhang, Wuhui Chen, Huawei Huang, Song Guo, Online computation offloading and traffic routing for UAV swarms in edge-cloud computing, IEEE Trans. Veh. Technol. 69 (8) (2020) 8777–8791.
[28] Mohamed-Ayoub Messous, Hichem Sedjelmaci, Noureddin Houari, Sidi-Mohammed Senouci, Computation offloading game for an UAV network in mobile edge computing, in: 2017 IEEE International Conference on Communications, ICC, 2017, pp. 1–6.
[29] Hongyue Kang, Xiaolin Chang, Jelena Mišić, Vojislav B. Mišić, Junchao Fan, Yating Liu, Cooperative UAV resource allocation and task offloading in hierarchical aerial computing systems: A MAPPO-based approach, IEEE Internet Things J. 10 (12) (2023) 10497–10509.
[30] Davide Callegaro, Marco Levorato, Optimal computation offloading in edge-assisted UAV systems, in: 2018 IEEE Global Communications Conference, GLOBECOM, 2018, pp. 1–6.
[31] Haitao Xu, Wentao Huang, Yunhui Zhou, Dongmei Yang, Ming Li, Zhu Han, Edge computing resource allocation for unmanned aerial vehicle assisted mobile network with blockchain applications, IEEE Trans. Wireless Commun. 20 (5) (2021) 3107–3121.
[32] Zhaolong Ning, Yuxuan Yang, Xiaojie Wang, Lei Guo, Xinbo Gao, Song Guo, Guoyin Wang, Dynamic computation offloading and server deployment for UAV-enabled multi-access edge computing, IEEE Trans. Mob. Comput. 22 (5) (2023) 2628–2644.
[33] Anandarup Mukherjee, Sudip Misra, Vadde Santosha Pradeep Chandra, Mohammad S. Obaidat, Resource-optimized multiarmed bandit-based offload path selection in edge UAV swarms, IEEE Internet Things J. 6 (3) (2019) 4889–4896.
[34] Wanning Liu, Yitao Xu, Nan Qi, Kailing Yao, Yuli Zhang, Wenhui He, Joint computation offloading and resource allocation in UAV swarms with multi-access edge computing, in: 2020 International Conference on Wireless Communications and Signal Processing, WCSP, 2020, pp. 280–285.
[35] Tan Do-Duy, Long D. Nguyen, Trung Q. Duong, Saeed R. Khosravirad, Holger Claussen, Joint optimisation of real-time deployment and resource allocation for UAV-aided disaster emergency communications, IEEE J. Sel. Areas Commun. 39 (11) (2021) 3411–3424.
[36] Hui Gao, Jianhao Feng, Yu Xiao, Bo Zhang, Wendong Wang, A UAV-assisted multi-task allocation method for mobile crowd sensing, IEEE Trans. Mob. Comput. 22 (7) (2023) 3790–3804.
[37] Sihem Ouahouah, Tarik Taleb, JaeSeung Song, Chafika Benzaid, Efficient offloading mechanism for UAVs-based value added services, in: 2017 IEEE International Conference on Communications, ICC, 2017, pp. 1–6.
[38] Junqin Huang, Linghe Kong, Guihai Chen, Min-You Wu, Xue Liu, Peng Zeng, Towards secure industrial IoT: Blockchain system with credit-based consensus mechanism, IEEE Trans. Ind. Inform. 15 (6) (2019) 3680–3689.
[39] Jiaxin Chen, Qihui Wu, Yuhua Xu, Nan Qi, Xin Guan, Yuli Zhang, Zhen Xue, Joint task assignment and spectrum allocation in heterogeneous UAV communication networks: A coalition formation game-theoretic approach, IEEE Trans. Wireless Commun. 20 (1) (2021) 440–452.



Runqun Xiong received the Ph.D. degree in computer science from Southeast University. He was with the European Organization for Nuclear Research as a Research Associate for the AMS-02 experiment from 2011 to 2012. He is currently an associate professor with the School of Computer Science and Engineering, Southeast University, China, where he is involved in AMS-02 data processing at the AMS Science Operations Center. His current research interests include cloud computing, industrial Internet, and drone-based wireless communication systems. He is a member of the ACM, IEEE, and the China Computer Federation.

Qing Xiao received his B.S. degree in Computer Science and Technology from Shanghai University in 2021. He is now a master student majoring in software engineering at Southeast University. His research interests include Blockchain, Edge Computing, and Internet of Things.

Zhoujie Wang received his B.S. degree in computer science from Nanjing University of Posts and Telecommunications, Nanjing, China, and received the M.S. degree in cyber science from Southeast University, Nanjing, China. He is currently a Software Engineer at Tencent Technology (Shanghai) Co., Ltd.

Zhuqing Xu received his B.S. degree from HangZhou DianZi University, China, and the M.S degree from the College of Software Engineering, Southeast University, China. He received the Ph.D. degree in computer science from Southeast University. His research interests include the Internet of Things, Cyber–Physical Systems, and Wireless Networks and Sensors.

Feng Shan received a Ph.D. degree in computer science from Southeast University, China, in 2015. He was a visiting student with the School of Computing and Engineering, University of Missouri-Kansas City, Kansas City, MO, USA, from 2010 to 2012. He is currently an Associate Professor with the School of Computer Science and Engineering, Southeast University. His research interests include the areas of Internet of Things, Wireless Networks, Swarm Intelligence, and Algorithm Design and Analysis.