Blockchain Empowered Secure Collaboration for Swarm Robots: Storage and Computation

Ran Wang, *Graduate Student Member, IEEE,* Sisui Tang, Hangning Zhang, Shihong Duan, Xiaotong Zhang, *Senior Member, IEEE,* Cheng Xu, *Member, IEEE*

***Abstract*—In recent years, swarm robot systems have garnered increasing attention, both in the industry and academia. These collaborative systems demand effective solutions for data storage, sharing, and security to unlock their full potential. To address these needs, this paper introduces a comprehensive distributed storage and computation framework based on blockchain and federated learning technology. The framework enables real-time collaborative data storage and computation, ensuring the security and reliability of collective intelligence systems. For data storage, we combine blockchain and dynamic containers to achieve secure and efficient storage of diverse robot data. To facilitate secure data utilization and sharing among robots, we present a federated learning-based collaborative computation approach. It allows robots to exchange model parameters while safeguarding data security, providing a versatile collaborative computation framework for collective systems. To validate the security and resilience of our framework, we present a practical scenario involving multi-agent collaborative localization. We conduct a thorough evaluation of the performance and security of this collaborative localization system, offering valuable insights for researchers in the field of swarm robotics.**

***Index Terms*—heterogeneous data, storage, byzantine fault tolerance, swarm robots, federated learning, blockchain.**

I. Introduction

As artificial intelligence (AI) technology continues to advance, there is an increasing need for collaboration among swarm robots, which consist of multiple agents [1]. These agents can take various forms, including physical robots, virtual robots, or other software agents. Using multiple agents, as opposed to a single agent, offers several advantages, such as increased robustness, fault tolerance, and flexibility. This makes them particularly well-suited for addressing challenges in hazardous, unknown, or dangerous environments, such as nuclear power plant collapses (e.g., Fukushima disaster) or environmental disasters like oil spills (e.g., Deepwater Horizon oil spill) [2]. Research in the collaboration of multi-agents has thus become a prominent direction to tackle such tasks.

The storage, sharing, and security of heterogeneous data collected by these multiple agents form a crucial foundation for their collaborative work. Each agent collects a wide range of data based on its surroundings and equipped sensors. These heterogeneous data can vary in terms of format, type, and quality. Within the domain of multi-agents, the widely used open-source Robot Operating System (ROS) [3] plays a significant role in swarm robotics systems. It employs a message-based communication mechanism that enables different nodes to communicate by publishing and subscribing to messages. This decoupling of dependencies among different modules makes the system more flexible and enhances fault tolerance [4]. Consequently, this paper builds upon ROS as a foundation and conducts research into the storage, sharing, and security aspects of heterogeneous robot data.

Nonetheless, ROS-based collective intelligence systems still confront several pressing issues related to the storage, sharing, and security of heterogeneous data. For instance, concerning robot data storage, ROS typically uses rosbag to store each message in a flat file (bag file). While this serves as a recording that can be played back, it is not suitable for more complex query tasks, such as searching for messages within specific time intervals or identifying timestamps when robots entered or exited specific states [5]. Regarding data sharing, a significant challenge arises due to the lack of global knowledge or explicit communication models among swarm robots. Traditional swarm robot systems heavily rely on local communication between neighboring robots and do not maintain global knowledge within the cluster [6]. In terms of security, there are concerns related to data theft, interception, or forgery. Adversaries could exploit the publish-subscribe pattern to steal stored data from ROS. Additionally, data transmitted between ROS nodes might be intercepted or manipulated [7], making it challenging to ensure data integrity during data retrieval from ROS.

Researchers have proposed various solutions to address these issues. Regarding ROS data storage, one potential solution is the availability of a utility that enables direct data storage in a database with advanced querying capabilities. This utility could simplify developers' access to relevant data through SQL-like data access languages, thereby mitigating the problem [8]. Other references [9], [10] introduce a NoSQL database into the ROS system for data storage, and benchmark tests demonstrate that their system performs on par with the native rosbag tool. However, most existing storage systems

Manuscript received XX XX 2025; revised XX XX 2025; accepted XX XX 2025. Date of publication XX XX 2025; date of current version XX XX 2025. This work is supported in part by the National Natural Science Foundation of China under Grant 62101029, in part by Guangdong Basic and Applied Basic Research Foundation under Grant 2023A1515140071, and in part by the China Scholarship Council Award under Grant 202006465043 and 202306460078. (*Corresponding author:* Cheng Xu)

The authors are with School of Computer and Communication Engineering, University of Science and Technology Beijing. Ran Wang, Shihong Duan, Xiaotong Zhang and Cheng Xu are also with Shunde Innovation School, University of Science and Technology Beijing (email: wangran423@foxmail.com; tangsisui@163.com; hangning0117@163.com; duansh@ustb.edu.cn; zxt@ies.ustb.edu.cn; xucheng@ustb.edu.cn).

in the literature serve as repositories for storing messages as binary files, lacking query functionality. To overcome these limitations, we propose a hybrid on-chain and off-chain storage approach for efficient and secure storage of heterogeneous data within the ROS system.

In the context of multi-agent data sharing and security, SROS [11] addresses security concerns in the data exchange process proposed by ROS through TLS and certificate mechanisms. In the ROS 2.0 phase, the integration of Data Distribution Services (DDS) [12] and SROS enhances the solution's authorization and access control security. However, ROS 2.0 has not yet implemented logging and data labeling. This means that when attackers gain central control, data can be directly tampered with without traceable error information. Consequently, a substantial body of research has focused on integrating blockchain technology into robot systems to establish the foundational framework for collaborative computation among multiple agents [13]–[15]. However, transaction data in blockchain-based collaborative computation frameworks is transparent to all participants, which doesn't fully ensure security during the sharing of sensitive data. As a result, the fusion of blockchain and federated learning technologies holds more promise in industrial settings [16], [17], facilitating transparent cooperation between organizations while safeguarding data privacy. Such a collaborative computation framework based on blockchain and federated learning is feasible for facilitating collaborative computation among robots without the need to share sensitive data. Therefore, we propose a multi-agent collaborative computation approach based on blockchain and federated learning. This approach not only supports flexible and efficient collaboration among multiple agents—where agents can easily join or exit collaborative computation tasks—but also ensures secure data sharing during the model training process, enhancing robustness and security in collaborative computation among multiple agents.

Taking these considerations into account, this paper introduces a multi-agent data security sharing framework based on blockchain and federated learning. This framework adopts a hybrid on-chain and off-chain storage approach to securely and efficiently store heterogeneous data from swarm robots. It implements a collaborative computation method based on federated learning, achieving secure data sharing among robots. Moreover, this paper aims to validate the performance and security of this framework in the context of a multi-agent collaborative localization application scenario. In summary, the primary contributions of this paper are as follows:

1) In the realm of collective intelligence systems, we propose a fully distributed framework for storage and computation. This framework aims to enable real-time collaborative data storage and sharing, with a strong emphasis on security and robustness.
2) To address storage challenges, we enhance ROS data storage by integrating blockchain with dynamic containers. This approach helps ROS manage its diverse data effectively while ensuring data integrity, auditability, and traceability through blockchain.
3) For efficient and secure data sharing among ROS nodes, we introduce a secure and efficient collaborative computation method based on federated learning. These methods offer a secure and versatile approach for collaborative tasks in multi-agent systems.
4) To validate the framework's practical application, we present a physical scenario using Ultra-Wideband (UWB) and Inertial Measurement Units (IMU) for autonomous navigation in conjunction with the Robot Operating System (ROS) for collaboration.

The rest of this paper is organized as follows: Section II discusses related research. Section III, IV and V explain the operational mechanisms of our multi-agent secure sharing federated learning framework. In Section VI, we assess the performance of the storage system, collaborative computation method, localization accuracy, and security within a collaborative localization scenario. Finally, Section VII summarizes the paper and discusses future directions.

II. Related Work

A. Data Storage in Swarm Robot Systems

A growing number of researchers are dedicating their efforts to developing effective database management systems for swarm robot systems, ensuring the proper storage of data. One common method for storing data in intelligent environments is the use of Relational Databases (RDBs) [18]. An illustrative example of such systems is the Monitoring System Toolkit by Lumpp et al. [7], which relies on MySQL for data storage. They provide a specific database design instance and conduct performance testing. Nevertheless, conventional relational database storage systems often struggle to meet the increasing scalability, availability, and real-time requirements of growing applications. As a result, researchers are increasingly exploring NoSQL databases as a means of storing data from swarm robot systems, considering them advantageous for handling heterogeneous data from multiple sources [19].

The introduction of NoSQL databases has led to the development of new data storage solutions. Implementing NoSQL in the field of robotics allows for the transformation of all acquired data into a usable format. Cuadros et al. [20] created an integration between ROS and MongoDB, supported by benchmark testing. Their research demonstrates comparable performance to the native rosbag tool with improved CPU and memory utilization. However, their paper does not attempt to determine the maximum throughput their system can support. Furthermore, the MongoDB system they discuss solely serves as a data repository that stores messages as binary files, lacking retrieval capabilities.

However, most of the previously mentioned storage systems primarily revolve around centralized storage, relying on central servers. This approach brings about issues related to single points of failure and storage efficiency. In current research, an increasing number of scholars are putting forward distributed storage solutions for swarm robot systems [21], [22]. However, these storage schemes lack an efficient retrieval approach and doesn't fully leverage the stored robot data. In the industrial domain, several studies have incorporated blockchain into secure big data sharing frameworks [23], [24]. By capitalizing on the distributed ledger characteristics of blockchain, shared

data is stored on the blockchain. This approach not only provides a decentralized, node-data synchronous, and secure storage architecture but also ensures the integrity of stored data. Therefore, the integration of blockchain into swarm robot systems offers an efficient and secure solution for storing and retrieving heterogeneous data from multiple sources.

B. Blockchain-Based Swarm Robots

Blockchain systems offer three key characteristics that make them well-suited for distributed multi-robot systems. First, their inherent security enhances data sharing within network systems [25], [26]. Second, their immutability allows for data auditing and tracing. Lastly, consensus protocols supported by smart contracts facilitate collaborative decision-making [27], [28]. In recent years, a growing number of researchers have integrated blockchain technology into swarm robot research. For instance, Kapitonov et al. [29] propose a protocol for autonomous drones integrated with ROS and the Hyperledger Fabric blockchain. Their architecture and implementation open up possibilities for innovation in various industries, including defense and agriculture. Similarly, Li et al. [30] introduce a blockchain-based collaborative edge knowledge inference framework to address trust issues during knowledge sharing in multi-robot systems. Salimi et al. [31] suggest a framework that integrates ROS 2 with Hyperledger Fabric blockchain for distributed robot systems, demonstrating its feasibility in inventory management applications involving ground and aerial robots. Grey et al. [32] introduce and implement the Swarm Contract framework, which leverages blockchain technology and employs tokens as incentives for human-interacting robot agents to execute collaborative tasks. This approach ensures trust through a reward system and adjudication solution. Swarm Contracts are customized robot smart contracts that enable secure and decentralized communication and collaboration among participants with varying levels of trust and heterogeneity. However, the studies mentioned above overlook a critical issue in swarm robot systems: the problem of Byzantine robot attacks. The presence of a Byzantine robot within the system can pose challenges in achieving the final collaborative task.

C. Federated Learning for Swarm Robots

The integration of blockchain and federated learning is increasingly being applied in various industries and the Internet of Things (IoT) field. This combination not only addresses security concerns related to collaborative data sharing but also offers a decentralized and trustworthy solution for managing data and collaborating on models in distributed computing. For instance, Lu et al. [33] propose a scheme for data sharing in industrial IoT based on blockchain and federated learning. By using federated learning, data demanders can retrieve and compute data without it leaving their local environment. However, this scheme cannot guarantee the integrity of the global model. Pokhrel et al. [16] design a Blockchain-based Federated Learning (BFL) system for privacy-conscious and efficient communication in vehicular networks. However, this design involves higher computational costs, leading to increased overall latency. Chai et al. [34] introduce a hierarchical blockchain-based federated learning algorithm for knowledge sharing. This hierarchical blockchain framework enhances reliability and security in knowledge sharing while also improving computational efficiency. Nevertheless, it faces challenges related to excessive storage resource consumption.

However, in the context of swarm robots, there is limited literature on the integration of blockchain and federated learning techniques for data sharing among swarm robots. Majcherczyk et al. [22] propose a distributed federated learning framework based on the Gossip protocol for spatiotemporal prediction in multi-agent systems. However, compared to blockchain, the Gossip protocol's distributed network raises concerns about security and decision consistency. As the cluster size grows, latency also increases, eventually affecting data sharing consistency among robots. Yu et al. [35] analyze the roles of distributed edge technology and federated learning in autonomous robot systems in their work, introducing key background concepts and considerations in current research, but they do not provide a specific solution.

III. The Proposed Framework

This section introduces a blockchain-based framework designed for secure data sharing among multiple agents. We will describe the components of the framework and its overall functioning mechanism. Additionally, we will delve into the storage system that supports collaborative tasks among multiple agents, effectively addressing storage issues within ROS. Furthermore, we will outline a collaborative computation method based on federated learning.

As shown in Fig. 1, each ellipsoidal dashed circle represents a robot, and all components are housed within a Docker. These components include the following:

Blockchain Node. The blockchain nodes of all robots create a blockchain network connected through a P2P network protocol for data transmission. Each blockchain node can initiate upload transactions to store and obtain the metadata of the robot's original data and globally updated weight values. The blockchain's consensus mechanism ensures the consistency of aggregated weight values received by all robots.

Local Model Training. Each robot trains a local model using a machine learning algorithm suitable for its sample set, updating local weight values through optimization algorithms. In this paper, we employ a PyTorch-based Long Short-Term Memory (LSTM) network for local model training, using Stochastic Gradient Descent (SGD) for weight value updates.

Robot Controller. This component acts as the "middleware" for each robot, facilitating interactions with various Docker components. It primarily provides two control interfaces: 1) The ROS system stores the acquired robot data in the data storage subsystem (DSS). The robot control program retrieves historical data from the DSS and transfers it to PyTorch as local input data for training; 2) Bidirectional data transfer interfaces facilitate communication between blockchain nodes and PyTorch, allowing local updated weight values to be sent to the blockchain and returning globally updated weight values to PyTorch.

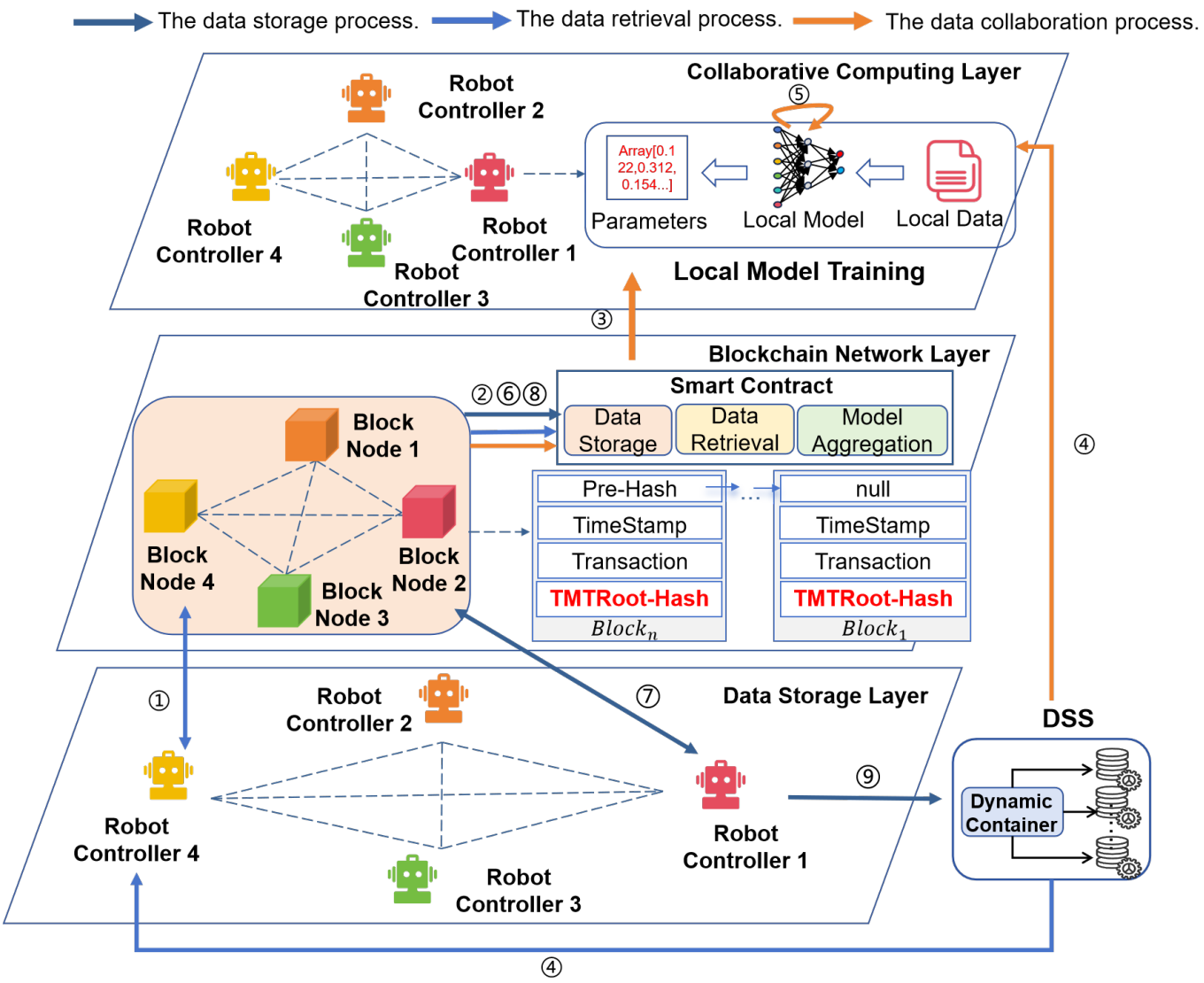


Figure 1: The framework diagram of multi-agent data security sharing.

Data Storage Subsystem (DSS). The off-chain data storage subsystem, designed using dynamic containers, acts as the data storage and management component for multi-agents. It handles the storage of heterogeneous data. The dynamic container allows for the construction of data structures from various types of robot data and enables the storage, packaging, and exchange of data using customizable templates suitable for data structures. In this storage system, backend users must define suitable data structures for different data types to support efficient storage operations. Further details about the dynamic container will be explained in Section IV-A.

Operation Mechanism. This framework leverages the decentralized nature of blockchain to create a distributed computing environment for multi-agent collaborative tasks. Any robot can flexibly join or exit the blockchain network. Smart contracts provide two essential functions for robots: data upload and data retrieval. At the outset of each collaborative task, initial weights for model training are established on the blockchain, acquired through random initialization specific to our PyTorch model. Weight values for local model updates are securely stored on the blockchain, ensuring data integrity and enabling blockchain nodes of all robots to retrieve and utilize updated weight values for optimizing local models. The primary workflow of this framework unfolds as follows:

1) The Robot Controller initiates the data retrieval function of the smart contract, transmitting a retrieval request to the blockchain to obtain the latest weight values ①②.

2) The blockchain returns the most recent aggregated weight values to the Robot Controller for local model updates ③.

3) The Robot Controller supplies locally stored original datasets and the latest aggregated weights from the blockchain as input parameters to PyTorch for local model training ④.

4) PyTorch constructs deep networks for local training and subsequently returns the trained weight values to the Robot Controller ⑤⑥.

⑤ The Robot Controller invokes the data upload function of the smart contract to transmit the updated local weight values, which are then employed within the smart contract to execute rolling averages and compute the aggregated weight values.

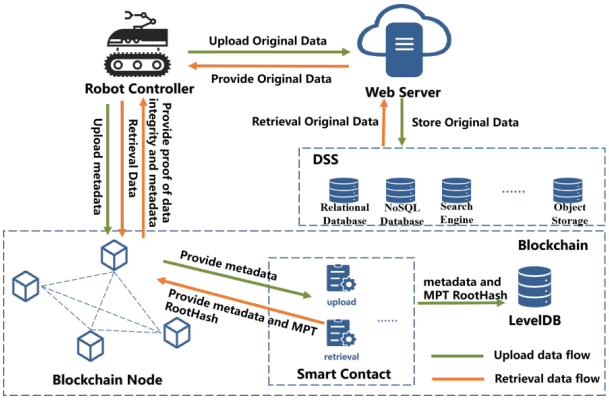


Figure 2: The architecture for heterogeneous robot data storage includes various components for data categorization, retrieval, and analysis.

Finally, the updated local weight values and aggregated weight values are securely stored on the blockchain ⑦-⑨.

IV. Storage Architecture for Heterogeneous Robot Data

The storage architecture of this framework is illustrated in Fig. 2. Given the sensitive and voluminous nature of most robot data, storing all data on a resource-limited blockchain would consume significant resources and pose the risk of data leakage. Therefore, considering privacy concerns and storage limitations, we adopt a hybrid approach, where we utilize the blockchain for data management. Specifically, we employ the strategy of "on-chain storage of transaction data, off-chain storage of raw data."

During the storage process, raw data is placed in a local off-chain storage system. This approach alleviates the computational burden on the blockchain and enhances the throughput of this framework. When it comes to data retrieval, blockchain-based verifiable retrieval mechanisms prevent data from being tampered with during storage and transmission, ensuring the integrity of the retrieved data. Simultaneously, records of uploaded updated weight values activities are securely stored on the blockchain, ensuring transparency and security throughout the entire federated training process. This approach not only achieves data tamper resistance but also provides traceability and auditability.

A. *Off-Chain Storage → Dynamic Containers*

Robot datasets often come in various formats, necessitating standardization into universal templates for accurate retrieval and analysis. This simplifies the user experience and reduces the learning curve. This section focuses on off-chain storage solutions for handling diverse robot data, encompassing data categorization, dynamic container composition, and the creation of containerized datasets.

(1) Categorizing Robot Data

Within ROS nodes, registration information is stored for topics and services. Each topic corresponds to a thread responsible for converting ROS messages into JSON formats.

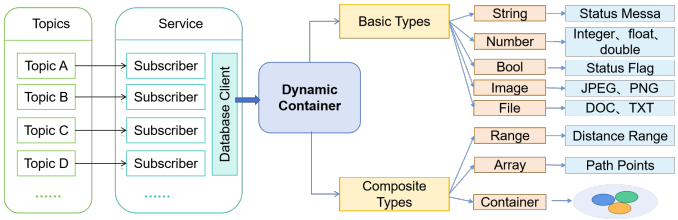


Figure 3: An example diagram of heterogeneous data structure for swarm robots.

These JSON messages are subsequently parsed into various types of data and stored in specific databases using dynamic containers. Fig. 3 illustrates the heterogeneous data structure within ROS nodes.

Designing dynamic containers that can efficiently store diverse robot data requires an analysis of this data's characteristics. This analysis involves identifying different types of data entities corresponding to topics published by ROS nodes. These entities may include sensor data, robot status, and more. For each data entity, relevant fields are defined to store specific information. For instance, a sensor data entity might include fields like timestamps, sensor types, and data values. The definition of these fields should align with the data types and application requirements. In some cases, data entities may have nested structures. For example, an image data entity could contain fields for pixel data, image dimensions, and image formats. To accommodate such complexity, nested document structures can be employed to represent these fields, enabling the storage of intricate data structures.

After analyzing the characteristics of robot data, this paper categorizes robot data into two main groups: basic types and composite types, as shown in Fig. 3. Basic Types include string, numeric, boolean, image, and file types. Composite Types are combinations of different basic types, including range, array, and container types. Range Types are formed from numeric types and represent an interval between two numeric values. Array Types consist of a single basic type, denoted as T, and represent an ordered list of values. Container Types can hold a value from any type within the set.

(2) Dynamic Container Composition

A dynamic container-based method for managing heterogeneous data is invaluable when dealing with large data volumes, diverse data types, and the need for rapid data retrieval and processing. In this article, we employ go-rosbag from GitHub to parse ROS system-generated bagfiles [36]. These files are specially formatted to store ROS messages with timestamps. We parse these files into corresponding data structures and store the parsed data as raw datasets in the Data Storage Subsystem (DSS). The DSS utilizes dynamic containers to store data with varying structures in different databases.

To accommodate the storage of heterogeneous robot data from multiple sources, we have developed a Dynamic Container Model (DCM) within the DSS. In everyday life, a container typically refers to a device used for storing, packaging, and transporting products, usually having a fixed internal structure for securing different types of products. In contrast, the abstract container integrated into DCM is designed to

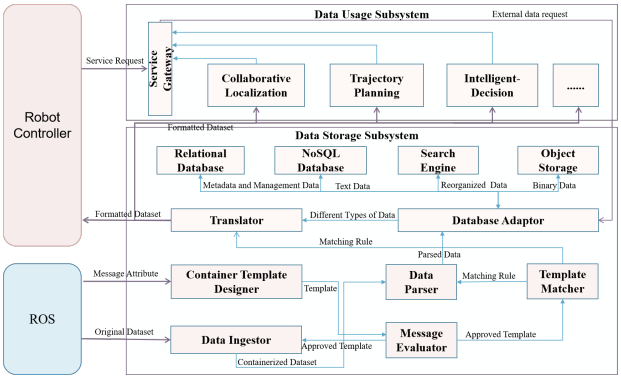


Figure 4: Architecture diagram of the multi-source heterogeneous robot data storage subsystem based on dynamic containers.

be dynamically constructed by various types of robot data. Therefore, DCM offers a means to store, package, and exchange data, allowing users to create templates suitable for data structures.

DCM comprises the following components: data collector, container template designer, template evaluator, template matcher, and data parser. Users can customize templates through the container template designer and submit them for review by the template evaluator. The data collector takes the raw dataset uploaded by the user and uses approved templates to normalize and convert it into a containerized dataset. The data parser and pattern matcher dissect the containerized dataset into metadata, textual robot data, binary files, etc., which are stored in the appropriate databases by database adapters. This standardized dataset simplifies subsequent retrieval and computational analysis of robot data. When the underlying database needs to provide retrieved raw data to the Data Usage Subsystem (DUS), different structures of robot data are reassembled and presented in a formatted manner based on template types, using translator matching template rules. The relationship diagram of various components in the dynamic container is shown in Fig. 4.

(3) The Creation of Containerized Datasets

Containerized datasets comprise two primary components: the container template and the container instance. The container template serves as an abstract description of the properties and structure within a robot dataset. We represent the relationship between a property and its data type using ":". A type declaration expression takes the form "$x : T$," where it signifies that property x holds data type T. A container template S includes a set of data-type declaration expressions, as shown below:

$$S = \{x_i : T_i^{i=\{1,\cdots,n\}}\} = \{x_1 : T_1, \cdots, x_n : T_n\} \quad (1)$$

Here, x_i corresponds to the properties, while T_i denotes the data type. These container templates offer support for custom attributes and structures. In essence, users enjoy the freedom to choose attribute names without any restrictions, but attribute values must conform to data types. The structure can be tailored by combining different data types.

On the other hand, a container instance represents a concrete description of the aggregated data. It defines the value of each property and adheres to the dataset template. An assignment expression like "$x = v$" indicates that property x is equal to v at a specific point. Consequently, the container instance C can be portrayed using a series of assignment expressions:

$$C = \{x_i = v_i^{i=\{1,\cdots,n\}}\} = \{x_1 = v_1, \cdots, x_n = v_n\} \quad (2)$$

A standardized representation of a robot dataset can be described as a containerized set denoted as (S, D). This set comprises a template and multiple instances, where $D = \{C_i^{i=1,\cdots,n}\} = \{C_1, \cdots, C_n\}$. Here, the dynamic container employs the template to normalize and transform the original dataset into a containerized format. This standardized dataset greatly simplifies the subsequent retrieval and computational analysis of robot data.

The design of dynamic containers enables efficient storage and management of diverse robot data. The adaptable template design of dynamic containers can easily accommodate the storage needs of various data types, enhancing data query efficiency and scalability. Depending on specific application requirements and data characteristics, the template design can be further optimized and extended to meet the system's performance and functional demands. Its modular nature ensures compatibility across heterogeneous robot systems, enabling seamless integration and efficient data standardization for retrieval and computation.

B. On-chain Storage → Template Merkle Trie (TMT)

To ensure secure data management at every stage of the data lifecycle, this framework leverages blockchain technology to store the metadata of raw data. This approach enables comprehensive data auditing, ensuring both data integrity and availability. To better accommodate the characteristics of multi-source heterogeneous robot data, we introduce the Template Merkle Trie (TMT) within the blockchain. The TMT design combines Merkle trees and Merkle Patricia Tries (MPT), producing template trees ($T_x - MPT$) that correspond to different topics (represented as x). All Tx-MPTs are constructed using the Merkle tree construction method, leading to the computation of the TMTRootHash, which is then stored in the block header. The structure of the TMT is depicted in Fig. 5.

Now, let's clarify the workflow of our blockchain-based data security storage framework. This process involves five main entities: ROS, Robot Controller, blockchain, LevelDB, and off-chain storage system. Within ROS's client interface, robot data templates are created and data forms are filled out, including raw data and corresponding metadata. The Robot Controller acts as middleware, serving as an intermediary for transaction requests. It isolates the blockchain from the off-chain storage system, ensuring the security of the underlying database. The blockchain is responsible for storing metadata corresponding to keywords and the addresses of raw databases. It creates data storage transactions for subsequent data traceability. LevelDB is utilized for the underlying storage of the blockchain, used to store and update the TMT structure. The off-chain storage system is responsible for storing all raw data.

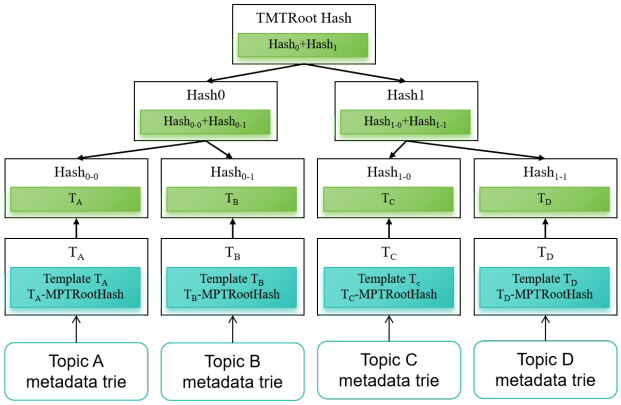


Figure 5: The TMT data structure.

The blockchain-based data storage process involves the following steps:

① *ROS Interaction*: Within ROS's client interface, robot data templates are selected or created based on topic types, data forms are filled out, and submissions are made.

② *Data Upload Request*: ROS initiates a data upload request to the Robot Controller.

③ *Robot Controller Processing*: The Robot Controller hashes the raw data and sends metadata (MD), the content hash of raw data, and its own signature (Sign) to the blockchain.

④ *Blockchain Verification*: The blockchain verifies the validity of the Robot Controller's signature. If successful, a data upload transaction is created.

⑤ *Transaction Broadcast*: Upon successful data upload transaction, it is broadcasted to the Robot Controller.

⑥ *Data Transfer*: After receiving the successful upload result, the Robot Controller sends the Content to the off-chain storage system.

⑦ *Keyword Extraction*: The off-chain storage system saves the raw data and uses the ElasticSearch (ES) algorithm [37] to extract keywords, creating a mapping between "keyword (key)" and "database storage address (DbA)."

⑧ *Mapping Return*: The "key-DbA" mapping is returned to the Robot Controller, indicating that data storage is complete.

⑨ *Storage Confirmation*: Upon receiving the storage completion message, the Robot Controller informs ROS that data storage is complete. It stores key, MD, and DbA in TMT format to LevelDB.

⑩ *TMT Maintenance*: The LevelDB database maintains and updates the TMT structure.

V. Secure Collaboration Based on Blockchain and Federated Learning

Building upon the architecture for storing multi-source heterogeneous robot data, we propose a collaborative computing method based on federated learning to effectively utilize robot data and facilitate collaborative tasks among multiple agents. This method allows for collaborative computing tasks without sharing the original sensitive robot data.

A. Local Model Training

It's worth noting that local models can be designed to meet specific application requirements, and their structure and complexity can vary significantly depending on their intended use. In this paper, we use LSTM network for local weight parameter training, and the training process is shown in Fig. 6.The following section offers a general description of the model's input, output, and learning objectives, along with specific examples from experimental settings.

We employ the mean squared error as the loss function, which is represented by the following formula:

$$\mathcal{L}_k(w) = \frac{1}{n_{s_k}} \sum_{i=1}^{N} \ell_i(x_i, y_i; w) \tag{3}$$

where N_s represents the total number of samples from all participating robots in aggregation, n_{s_k} represents the number of samples for the k-th robot, w represents the model's weight parameters, $\ell_i(x_i, y_i; w)$ represents the loss obtained by predicting the sample (x_i, y_i) with model parameters w, and x_i and y_i represent the i-th training data point and its corresponding label.

The optimization algorithm used for weight updates is Stochastic Gradient Descent (SGD). Each robot calculates the gradient $g_k^t = \frac{\partial \mathcal{L}_k(w_k^t)}{\partial w_k^t}$ of the loss function $\mathcal{L}_k(w_k^t)$ in the current training round. Then, the k-th robot updates its local model parameters w_k^{t+1} using the initialized global model parameters w_k^t and the gradient g_k^t:

$$w_k^{t+1} = w_k^t - \eta g_k^t, \quad \forall k \in \{1, 2, ..., 7\} \tag{4}$$

where η represents the learning rate.

After completing the update of local model parameters, each robot passes the updated weights to the blockchain network for parameter aggregation by calling the smart contract's submitWeights function.

B. Model Aggregation

When weights are submitted to the blockchain, rolling average is performed to reduce computational complexity. Only two lists are stored on the blockchain: latestWeights and totalWeights. Here are the formulas for calculating them:

$$totalWeights_i^{t+1} += n_{s_k}^{t+1} \cdot w_{k,i}^{t+1}, \quad \forall i \in N \tag{5}$$

$$latestWeights_i^{t+1} = \frac{totalWeights_i^{t+1}}{N_s^{t+1}}, \quad \forall i \in N \tag{6}$$

where $n_{s_k}^{t+1}$ represents the number of samples used for training robot k in round $t+1$, $w_{k,i}^{t+1}$ is the weight value of robot k after the $t+1$-th round of training, and N_s^{t+1} is the total number of samples in round $t+1$ (sum of all $n_{s_k}^{t+1}$).

At present, there are different aggregation methods used in federated learning, including FedAwo [38], FedProx [39], and FedNova [40]. However, in this paper, we primarily employ the Federated Averaging algorithm (FedAvg) [41], which is a straightforward and efficient algorithm suitable for LSTM layers. The FedAvg algorithm is an optimization approach that resembles the original neural network objective. You can find

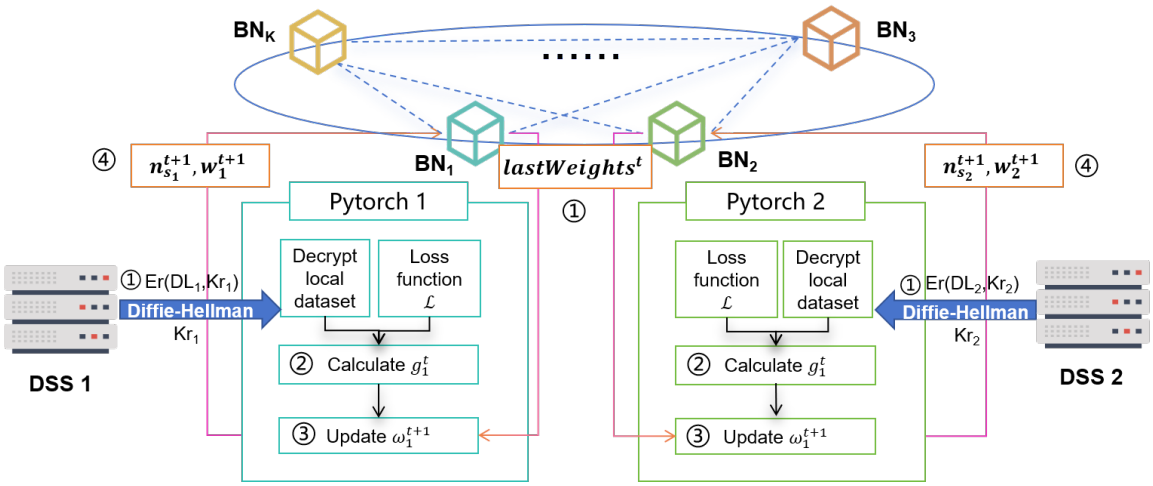


Figure 6: The process of local model training.

the detailed algorithm in Algorithm 1, and the formula for the loss function during the aggregation process is as follows:

$$\min_{w\in\mathbb{R}^d} \mathcal{L}(w) = \sum_{k=1}^{n} \frac{n_{s_k}}{N_s} \mathcal{L}_k(w) \tag{7}$$

where N_s represents the total number of samples from all participating robots in aggregation, n_{s_k} represents the number of samples for the k-th robot, w represents the model weights, and $\mathcal{L}_k(w)$ represents loss function of the k-th robot.

Algorithm 1 FedAvg

Input: Number of robots participating K, Local mini-batch size B, Local epochs E, Aggregation rounds R, Learning rate η

Output: Updated weight set w

Blockchain executes:
initialize $w(0)$
for round $t = 1$ to R **do**
  $p \leftarrow max(K, 1)$
  $S_t \leftarrow$ (random set of p robots)
  for robot $k \in S_t$ **do**
    $w^k(t+1) \leftarrow RobotTrain(k, w(t))$
  end for
  $w(t+1) \leftarrow \frac{1}{N} \sum_{k=1}^{p} n_k \cdot w^k(t+1)$
end for
RobotTrain(k,w):
$\mathcal{B} \leftarrow$ (split $\mathcal{P}_k$ in batches of size B)
for epoch $e = 1$ to E **do**
  for batch $b \in \mathcal{B}$ **do**
    $w \leftarrow w - \bigtriangledown \ell(b; w)$
  end for
end for
return w

C. Global Model Consensus

After the completion of model aggregation, a transaction will be generated. Miners will package this transaction into a block and broadcast it to the blockchain network, where consensus will be reached among the blockchain nodes of all robots. In this phase, a consensus mechanism is employed to determine the aggregated parameters that are accepted by all robots. The framework utilizes the PBFT algorithm [42], capable of tolerating up to f Byzantine fault nodes. To enhance the efficiency of the consensus mechanism, nodes are categorized as active nodes and passive nodes. The initiator of consensus, the miner, acts as the primary node. Robots participating in the aggregation process are referred to as active nodes, while those not engaged are labeled as passive nodes. The consensus mechanism encompasses five steps: request, pre-prepare, prepare, commit, and reply.

Request Phase: In this phase, the primary node initiates a global model consensus request to all active nodes within the blockchain network.

Pre-Prepare Phase: The primary node calculates the $Hash(totalWeights^{t+1})$ and broadcasts it throughout the entire blockchain network.

Prepare Phase: Upon receiving the $Hash(totalWeights^{t+1})$ sent by the primary node, active nodes individually calculate the hash value of their received $totalWeights^{t+1}$. If it matches the hash value sent by the primary node, they send confirmation messages to the blockchain network.

Commit Phase: After receiving confirmation messages from $2f$ active nodes, all nodes broadcast confirmation information of the Commit phase to other active nodes.

Reply Phase: When each active node receives confirmation messages from more than two-thirds of the participating consensus nodes, the consensus request task is completed. A reply message is constructed and sent back to the primary node. Upon receiving confirmation from over two-thirds of the nodes, the primary node establishes the global model and broadcasts the hash of the global model to all active and passive nodes for storage.

Upon the completion of the current round of model training, the process of local model training, global model aggregation, and global model consensus will be continually repeated until

Figure 7: Showcase of the experimental arena and equipment for physical validation experiments. **Left** depicts the layout of the experimental scene, with AutoRobot and Crazyflie initiating their movements from the randomly assigned positions marked by ▲ and ■ icons, respectively. Both vehicles converge towards the central location. The **Upper Right** and **Lower Right** figures showcase the AutoRobot unmanned vehicle and the Crazyflie unmanned drone, respectively, illustrating the composition of the heterogeneous system utilized in this experiment.

the loss function reaches the set threshold or the maximum training time is reached. Although PBFT offers strong Byzantine fault tolerance, its performance degrades significantly with increasing node numbers due to high communication complexity, which limits scalability in large swarm systems.

VI. Experimental Analysis and Discussion

In this section, we conducted testing and evaluation of a collaborative localization system using the designed framework in a real-world localization scenario. We have developed a container template design tool that facilitates users to modify or create new templates for topics. In terms of performance evaluation, we assessed the system's the performance of data storage, the loss rate in federated learning during the collaborative computation process, and the localization accuracy of the multi-agent system.

A. Experimental Setup

The localization scenario took place in a 10m × 10m physical area and involved two autonomous robots, AutoRobot [43], and two micro-drones, Crazyflie [44]. These robots were equipped with various Ultra-Wideband (UWB) and Inertial Measurement Unit (IMU) sensors with sampling frequencies of 200Hz and 10Hz, respectively, as depicted in Fig. 7. During the experiment, the robots moved randomly, and we collected sensor information and location data via ROS. Ground truth data was captured using the NOKOV optical motion tracking system [45]. The IMUs provided motion data for the entities, while UWB sensors measured distances between agents.

In this study, our blockchain environment setup consisted of three essential components: the blockchain network, device configuration, and testing tools. For the blockchain network, we initiated each node on the ROS within each agent as a Docker container, connecting them to the blockchain network via Docker Swarm. Additionally, we generated configuration files for the blockchain network, specifying parameters

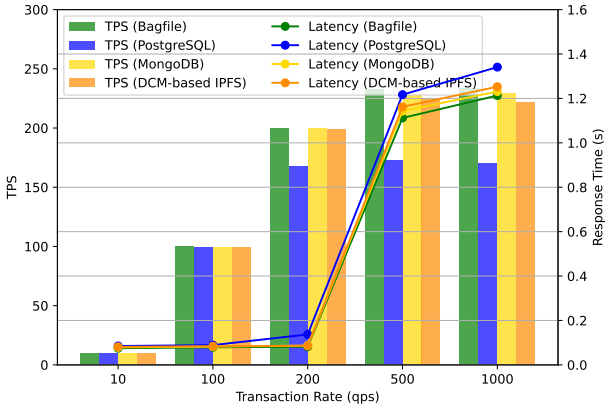


Figure 8: Performance comparison of different ROS data storage schemes.

like organizations, node types, channel names, and more. Regarding testing tools, we employed Jmeter to assess the blockchain's impact on the framework's performance. This involved simulating user-related requests using HTTP requests to evaluate the system's responsiveness.

During the federated learning process, PyTorch is considered to realize an LSTM local network for training weight parameters. The neural network architecture consists of the following components:

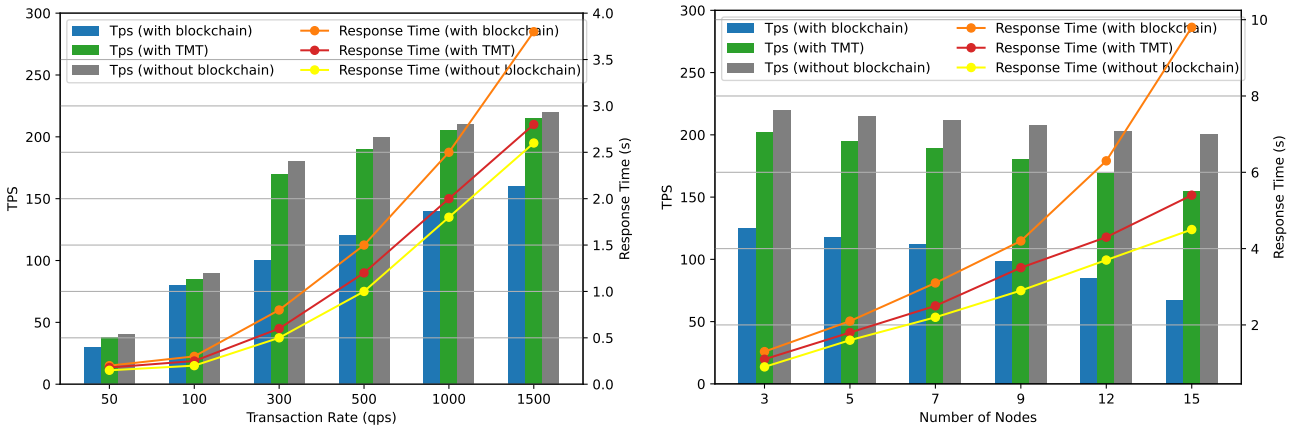
- First layer: m LSTM units with an input of $I_t \in \mathbb{R}^{n \times d}$.
- Second layer: Dropout layer with a specified rate of P.
- Third layer: Dense layer with N_{hidden} neurons.
- Final layer: Reshape layer, resulting in an output shape of $O_t \in \mathbb{R}^{h \times w}$, which corresponds to the output duration for predicting localization trajectories in our method.

In the first LSTM layer, m represents that the number of LSTM units is set to 16, and I_t is an input matrix being set to 32x2. It represents the number of samples in each training batch, where n is the length of the input sequence or the number of time steps, and d is the dimensionality or the number of features at each time step. P denotes the specified dropout rate for the Dropout layer, and set to 20%. In the third layer, N_{hidden} that represents the number of neurons is set to 96. The output matrix of the final layer is denoted as O_t being set to 48x2, representing the output parameters of the model training. The values of h and w in the matrix depend on the training task, context information, batch size, and other factors. We configured the batch size for training the local model to be 20, with 20 local iterations. The learning rate for the SGD optimization algorithm was set at 0.001.

B. Performance of Data Storage

In this section, we evaluated the performance of different storage schemes, the impact of blockchain on storage performance.

1) Performance Comparison of Different ROS Data Storage Schemes : In this experiment, we conducted a comprehensive comparison of the off-chain data storage subsystem, known as DSS, proposed in this article with several existing storage solutions. The solutions considered for comparison include



(a) (b)

Figure 9: The TPS and Latency of system data storage performance statistics. (a) Transaction rate (qps). (b) Number of nodes.

ROS's built-in bagfile [36], the relational database PostgreSQL solution [46], and the non-relational database MongoDB solution [47]. Our evaluation focused on two critical aspects: *storage throughput* and *response time*.

As depicted in Fig. 8, the comparison results reveal noteworthy insights. Firstly, in contrast to PostgreSQL, DSS demonstrates substantial improvements in both throughput and response time. This observation underscores the limitation of relational database storage solutions when handling the diverse and heterogeneous data generated by robots.

When compared to both bagfile and MongoDB, our DSS solution exhibits a minor gap in terms of throughput and response time. This discrepancy can primarily be attributed to the additional step in our solution, which involves the analysis and processing of data through dynamic containers before its storage in the database. However, it's crucial to emphasize that this difference is relatively insignificant and can be considered negligible in practical terms.

Furthermore, it's essential to highlight a distinct advantage of our solution. In contrast to bagfile and MongoDB, our approach offers superior flexibility and scalability in managing various data types across different databases. This capability is made possible through template design, ensuring that stored data is standardized and easily manageable, thereby enhancing the overall efficiency and effectiveness of data storage processes.

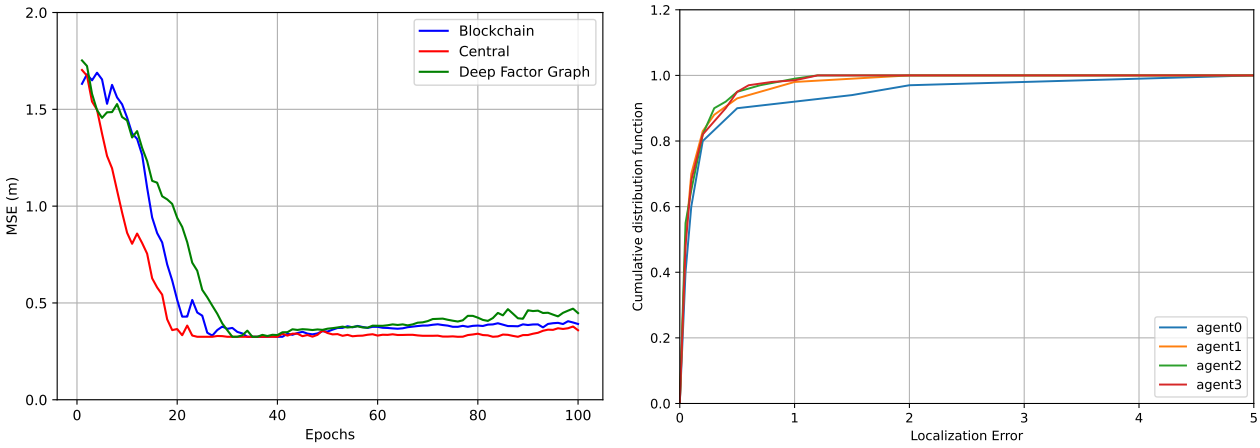
2) Impact of Blockchain on Data Storage Performance: ***During the data storage phase***, the primary performance bottleneck in data storage, compared to the storage performance of the off-chain data storage subsystem, is the blockchain. Therefore, in this experiment, we will compare and analyze the throughput and latency of the data storage framework before and after incorporating the blockchain. We will set transaction rates at 50, 100, 300, 500, 1000, and 1500 transactions per second (tps) and test the throughput and latency of data storage before and after adding the blockchain.

Integrating blockchain into the data storage framework introduces additional overhead compared to traditional off-chain storage, primarily due to consensus validation, cryptographic computations, and redundancy in data replication. As shown in Fig. 9-(a), the unoptimized blockchain-based storage system significantly reduces throughput and increases response time, with a maximum observed TPS of approximately 120, beyond which latency grows sharply due to transaction backlog. This is primarily caused by the increased computational load for transaction verification and block finalization, leading to noticeable performance degradation compared to the non-blockchain storage system.

To address this issue, we introduce the TMT (Template-Merkle-Patricia Trie) indexing structure, which improves storage efficiency by organizing data hierarchically based on material templates (as depicted in Fig. 5). By leveraging the combined benefits of Merkle trees and MPT structures, TMT significantly reduces the lookup and verification overhead, enabling higher throughput and lower latency compared to the unoptimized blockchain model. Although performance is still slightly lower than that of non-blockchain storage, the TMT-enhanced blockchain system achieves notable improvements in storage efficiency while maintaining the core security benefits of blockchain.

To evaluate the scalability of the proposed method with respect to the number of participating nodes, we tested the system's storage performance in terms of throughput (TPS) and response time under three configurations—standard blockchain, the improved TMT mechanism, and a baseline without blockchain—across varying numbers of agent nodes (3, 5, 7, 9, 12, and 15). As shown in Fig. 9-(b), the results show that as the number of nodes increases, the standard blockchain configuration experiences a noticeable decline in throughput and a sharp increase in response time. In contrast, the TMT-enhanced system exhibits a much more stable performance trend, with only moderate degradation, and remains close to the performance of the non-blockchain baseline.

Despite the slight performance drop compared to off-chain storage, the security, auditability, and traceability enhancements provided by blockchain outweigh the trade-offs. TMT indexing bridges the performance gap by optimizing transaction processing and retrieval efficiency, making blockchain-based storage a more viable solution for materials data man-



(a) (b)

Figure 10: The statistical results of multi-agent localization accuracy. (a) Loss curves for centralized, blockchain-integrated, and factor graph-based localization. (b) The cumulative distribution of errors for each agent.

agement. Additionally, while TMT improves lookup efficiency, it still introduces computational overhead for trie maintenance and hash calculations, potentially affecting real-time responsiveness under heavy workloads. Future optimizations, such as further indexing refinements or hardware acceleration, could further enhance performance while preserving the integrity and security advantages of blockchain technology.

C. Performance of Collaborative Computation

In this experiment, we evaluated the performance of our collaborative computation method by examining the loss during the model training process within the collaborative localization system.To demonstrate that our blockchain-based distributed framework enhances system security without compromising performance, we compared our method with both a centralized federated learning approach [41] and the state-of-the-art deep factor graph method [48] in the localization system.

As shown in Fig. 10-(a), our method's average loss converges as aggregation rounds progress. By the time we reach 80 aggregation rounds, the average loss stabilizes around 0.0187. Our method exhibits a relatively swift convergence rate in the initial 40 rounds of aggregation, resulting in a small final average loss. Although our method's convergence rate is slower compared to the centralized federated learning method, the use of a blockchain-based distributed computing environment does not hinder the ultimate convergence of the average loss. In comparison, the distributed deep factor graph-based localization algorithm shows an even slower convergence rate in the early stages, requiring more iterations to reach a comparable level of accuracy. This highlights the efficiency of our blockchain-integrated federated approach in balancing convergence speed and system decentralization.

For localization accuracy, we deployed four agents that began moving in different directions from various randomly assigned initial positions. These initial positions were set as unknown random values. Additionally, Fig. 10-(b) shows the cumulative error distribution for each agent. These results reveal that each intelligent agent has a probability of over 80% to reduce localization error to within 0.5 meters. In conclusion, our framework demonstrates high accuracy and robustness in multi-agent localization tasks. Even when initial positions are unknown, the intelligent agents swiftly converge to the correct trajectory with localization errors well-contained within a narrow range.

D. Scalability Evaluation

The experiment evaluates the impact of increasing the number of agent nodes on the accuracy of collaborative localization. As shown in Fig. 11, as the number of nodes increases, localization accuracy improves, with lower mean squared error (MSE) values. However, beyond a certain number of nodes, the accuracy gains become negligible, indicating that adding more agents beyond a threshold does not significantly enhance localization performance. This is likely because once sufficient collaborative information is available, additional nodes contribute redundant or overlapping information rather than meaningful improvements in accuracy.

Additionally, the convergence speed of the loss curves increases as the number of nodes grows. With more nodes participating in collaborative localization, the system achieves faster convergence, reaching stable localization accuracy in fewer epochs. This phenomenon can be attributed to the increased amount of shared observations and inter-agent corrections, which accelerate the refinement of position estimates. However, after reaching an optimal node count, further additions do not notably accelerate convergence, suggesting that the system reaches an equilibrium where additional agents offer diminishing contributions. These findings highlight the importance of selecting an appropriate number of agents to balance efficiency and accuracy in multi-agent localization tasks.

E. Multi-Adversarial Attack Analysis

In the context of federated learning-based multi-agent collaborative localization, three common types of adversarial attacks are considered:

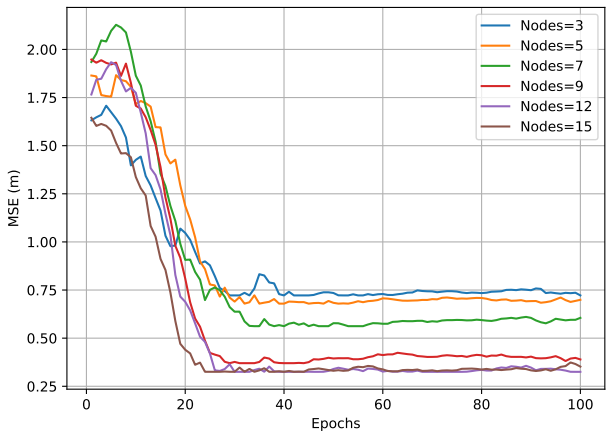


Figure 11: Comparison of localization accuracy loss curves under different numbers of nodes.

Noise Attack. The Byzantine nodes send noise perturbed gradients that generated by adding Gaussian noise into honest gradients [49]. We take the same Gaussian distribution parameters as random attack.

A Little is Enough. the Byzantine nodes send malicious gradient vector with elements crafted [50]. For each node $i \in [d]$, the Byzantine nodes calculate mean (μ_i) and standard deviation (σ_i) over benign updates, and set corrupted updates Δi to values in the range $(\mu_i - z^{\max}\sigma_i, \mu_i + z^{\max}\sigma_i)$, where $z^{\max}$ ranges from 0 to 1. We set $z^{\max} = 0.3$ in our experiment.

Inner Product Manipulation (IPM) Attack. The Byzantine nodes manipulate the inner product between the model updates and the global gradient direction to mislead the aggregation process [51]. Specifically, the malicious nodes generate adversarial updates by perturbing the gradient vectors such that their inner products with the true gradient direction are minimized or even reversed. This manipulation results in a slower convergence rate or divergence of the global model while remaining undetectable under standard anomaly detection techniques.

The experimental results in Fig. 12-(a) illustrate the impact of these attacks on localization accuracy. The Mean Squared Error (MSE) loss curves show that federated learning models demonstrate greater resilience to adversarial attacks compared to single-agent models. Here, the "Single Agent" curves represent the averaged localization accuracy across multiple agents in a scenario without the federated learning framework, where each agent updates its model independently based on local observations. Without aggregation, individual models are more vulnerable to attack-induced distortions, leading to significant fluctuations and a lack of convergence.

On the other hand, under Noise and IPM attacks, the FedAvg-based models still achieve convergence, albeit at a slower rate, demonstrating a level of robustness against these threats. However, the ALIE attack proves to be more disruptive, as its adversarial update strategy significantly degrades performance by introducing statistically biased gradients into the aggregation process. Despite this, FedAvg maintains a much lower overall error compared to single-agent models.

As shown in Fig. 12-(b), to demonstrate the algorithm's ability to resist multiple Byzantine nodes, we present the loss curves under Noise, ALIE, and IPM attacks when 7 out of 15 participating nodes are Byzantine. The results show that although nearly half of the nodes are adversarial, the model can still gradually converge to a low error level despite a slower convergence rate, ultimately achieving a similar level of accuracy to the scenario with fewer Byzantine nodes (see Fig. 12-(a)), demonstrating strong robustness against attacks. In contrast, the single-agent model fails to converge under the same attack conditions, with the error fluctuating at a high level. This phenomenon can be attributed to the fact that federated learning aggregates information from multiple agents, allowing the global model to retain useful patterns even in the presence of malicious updates. In comparison, the single-agent model relies solely on local information and lacks mechanisms to counteract adversarial interference, making it more vulnerable to divergence and misleading training trajectories.

These findings emphasize the need for robust aggregation strategies in federated learning to further mitigate adversarial impacts. Future research should explore more secure aggregation methods, such as MultiKrum [52], CenteredClipping [53], GeoMed [54], and Median [55], among others.. Additionally, integrating blockchain with federated learning for secure update verification, along with anomaly detection mechanisms and cryptographic techniques like differential privacy and secure multi-party computation, can further safeguard the collaborative localization process.

F. Discussion

The integration of blockchain and federated learning in multi-robot systems provides a promising approach to addressing key challenges in data privacy protection, computational load balancing, and network security. Blockchain's decentralized trust mechanism, combined with federated learning's distributed model training, enhances security, efficiency, and scalability in collaborative robotic environments. The following sections discuss how our approach effectively tackles these challenges and outline future directions for improvement.

1) Data Privacy Protection: One of the key challenges in multi-agent collaborative learning is data privacy protection. Robots operating in different environments often collect highly sensitive or proprietary data that cannot be shared directly. Federated Learning (FL) provides a certain level of privacy protection by enabling local model training and sharing only model updates. However, traditional federated learning frameworks still face various privacy threats. The decentralized nature of blockchain mitigates the risk of privacy breaches associated with traditional federated learning, which relies on a single centralized aggregation server.

In our proposed method, blockchain is used to store the encrypted hash values of model updates rather than the actual data, ensuring that no private information is exposed during transmission. This mechanism prevents data tampering and unauthorized modifications, as all updates must undergo PBFT (Practical Byzantine Fault Tolerance) consensus verification

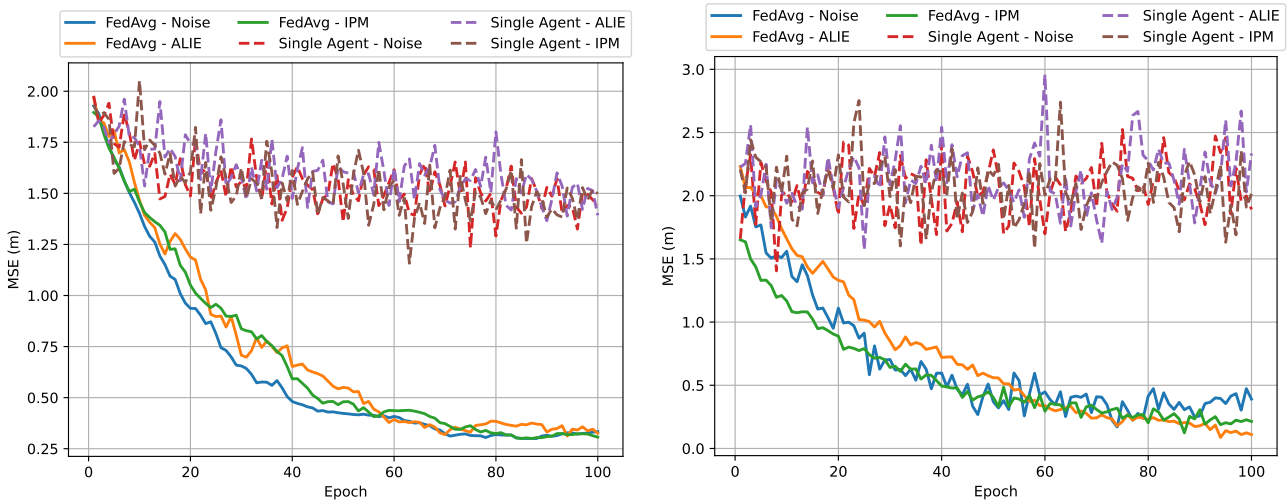


Figure 12: Comparison of loss curves under three different attack scenarios. (a) Real-world scenario: 4 agent nodes with 1 byzantine node. (b) Simulation scenario: 15 agent nodes with 7 byzantine nodes.

before being accepted. By leveraging blockchain's immutability, decentralized trust mechanism, and cryptographic security, our approach effectively enhances privacy protection in federated learning while ensuring model training efficiency and transparency.

Furthermore, our proposed method can be combined with differential privacy, homomorphic encryption, and secure multiparty computation (SMPC) to further enhance privacy protection in the future. This integration ensures that even if an attacker gains access to the gradient update records stored on the blockchain, they will not be able to reconstruct the original data. Consequently, our method provides resistance against model inversion attacks, membership inference attacks, and gradient leakage risks, further strengthening the security of the federated learning process.

2) Computational Load Balancing: In large-scale multi-robot systems, efficient allocation of computational resources is a critical challenge. Due to variations in computational power, battery life, and network stability across different robots, traditional federated learning methods often assume equal computational capabilities among all participants, which is impractical in real-world applications. The introduction of blockchain provides a decentralized task coordination mechanism while optimizing data retrieval and model aggregation complexity, thereby improving computational load balancing.

One of the key optimizations in our method is the introduction of the Template Merkle Trie (TMT) indexing structure, which significantly enhances blockchain-based query operations. In conventional blockchain storage systems, querying and verifying historical model updates have a computational complexity of $O(n^2)$ due to the need to sequentially traverse all block records. By employing the TMT structure, we reduce the query complexity to $O(log(n))$, enabling faster data access and verification. This optimization significantly reduces the computational burden on resource-constrained robots when retrieving data from the blockchain, while ensuring the integrity of federated learning updates. Additionally, smart contracts can be utilized for dynamic training task allocation, adapting to each robot's computational capacity instead of enforcing a uniform synchronous aggregation of updates.

Building on existing research, our method will support asynchronous federated learning (AFL) in the future, allowing robots with limited computational resources to participate in training at their own capacity without affecting global model convergence. Furthermore, we plan to adopt model pruning and knowledge distillation techniques to reduce the size of model parameters, thereby lowering computational and communication overhead. This approach ensures that low-power devices can still contribute effectively while preventing overall training delays caused by slower nodes.

3) Network Security: In the federated learning process of multi-robot systems, communication between robots involves the exchange of model updates, which may be vulnerable to model poisoning, Sybil attacks, and data tampering threats. The integration of blockchain enhances network security by providing an immutable distributed ledger that records transparent audit logs of all critical transactions in the federated learning process. By enforcing consensus-based verification mechanisms, blockchain ensures that only legitimate model updates are incorporated into the training process, effectively mitigating the risks posed by malicious attacks.

A key security enhancement in our approach is the Byzantine fault-tolerant model aggregation mechanism, which has been validated through experimental results. In a federated learning environment, some robots may attempt to submit malicious gradient updates to mislead the global model. To address this issue, we adopt PBFT consensus, which requires at least two-thirds of the nodes to reach agreement before accepting a model update. This mechanism effectively prevents adversarial nodes from corrupting the global model and ensures that attackers cannot manipulate the learning process through falsified gradients. Our experimental results

demonstrate that even when a certain proportion of robots exhibit Byzantine behavior, the federated learning process maintains model integrity.

Looking ahead, we will further investigate trust-aware federated learning, in which robots are assigned a credibility score based on their historical behavior, and higher-weighted contributions are prioritized during model aggregation to reduce the impact of malicious robots. Additionally, we plan to incorporate Zero-Knowledge Proofs (ZKP) and Secure Enclaves (SGX) to further enhance the security and privacy protection of federated learning. These enhancements will ensure that our method remains robust even in highly adversarial environments, enabling secure and efficient multi-robot collaborative learning.

VII. Conclusion and Prospect

In response to the demands of swarm robotics applications, by utilizing a blockchain network, our proposed framework establishes a decentralized and trusted platform for swarm robots, significantly enhancing the efficiency, security, and reliability. In the terms of data storage, addressing the scalability and security challenges faced by ROS, we propose a framework for storing heterogeneous robot data that combines blockchain and dynamic containers. The flexible template design of dynamic containers is well-suited to the storage needs of diverse data, ensuring efficient storage and management of robot data. Simultaneously, blockchain stores metadata for robot data, providing the collective intelligence system with data integrity, auditability, and traceability capabilities. Moreover, the metadata stored on the blockchain allows for retrieval and computation. The collaborative computation approach introduced here, based on federated learning, leverages data distributed among swarm agents for model training.

To evaluate the security and robustness of this framework, we applied it to real-world collaborative localization scenarios. The experimental results confirm that the framework meets the practical requirements of collaborative localization systems in terms of data storage, collaborative computation, and security. It provides an effective solution to the challenges faced by swarm systems concerning data storage, sharing, and security.

References

[1] M. Mashayekhi, N. Ajmeri, G. F. List, M. P. Singh, Prosocial norm emergence in multi-agent systems, ACM Transactions on Autonomous and Adaptive Systems (TAAS) 17 (1-2) (2022) 1–24.

[2] J. Wang, Y. Hong, J. Wang, J. Xu, Y. Tang, Q.-L. Han, J. Kurths, Cooperative and competitive multi-agent systems: From optimization to games, IEEE/CAA Journal of Automatica Sinica 9 (5) (2022) 763–783.

[3] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, W. Woodall, Robot operating system 2: Design, architecture, and uses in the wild, Science robotics 7 (66) (2022) eabm6074.

[4] S. Zhang, M. Tang, X. Li, B. Liu, B. Zhang, F. Hu, S. Ni, J. Cheng, Rosethereum: A convenient tool to bridge ros and blockchain (ethereum), Security and Communication Networks 2022 (1) (2022) 7206494.

[5] F. Duan, W. Li, Y. Tan, The framework and fundamental use of ros, in: Intelligent Robot: Implementation and Applications, Springer, 2023, pp. 43–69.

[6] V. Varadharajan, D. St-Onge, B. Adams, et al., Soul: Data sharing for robot swarms, Autonomous Robots 44 (3-4) (2020) 377–394.

[7] F. Lumpp, M. Panato, N. Bombieri, et al., A design flow based on docker and kubernetes for ros-based robotic software applications, ACM Transactions on Embedded Computing Systems 23 (5) (2024) 1–24. doi:10.1145/3643721.

[8] Y. Ogiwara, H. Kawashima, Extending ros transform library for massive autonomous robots, in: Proceedings of the 2023 IEEE 29th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA), IEEE, 2023, pp. 277–278. doi:10.1109/RTCSA58093.2023.00054.

[9] R. Kabir, Y. Watanobe, K. Nakamura, R. Islam, K. Naruse, An efficient cloud framework for multi-robot system management, in: New Trends in Intelligent Software Methodologies, Tools and Techniques, IOS Press, 2021, pp. 605–617.

[10] L. Fernández-Becerra, Á. M. Guerrero-Higueras, F. J. Rodríguez-Lera, C. Fernández-Llamas, Analysis of the performance of different accountability strategies for autonomous robots, in: 14th International Conference on Computational Intelligence in Security for Information Systems and 12th International Conference on European Transnational Educational (CISIS 2021 and ICEUTE 2021) 14, Springer, 2022, pp. 41–51.

[11] N. Goerke, D. Timmermann, I. Baumgärt, Who controls your robot? an evaluation of ros security mechanisms, in: Proceedings of the 2021 7th International Conference on Automation, Robotics and Applications (ICARA), IEEE, 2021, pp. 60–66. doi:10.1109/ICARA51699.2021.9376520.

[12] Q. Lu, X. Li, Y. Guan, et al., Modeling and analysis of data flow-oriented ros2 data distribution service, International Journal of Software & Informatics 11 (4).

[13] P. Simoens, M. Dragone, A. Saffiotti, The internet of robotic things: A review of concept, added value and applications, International Journal of Advanced Robotic Systems 15 (1) (2018) 1729881418759424.

[14] J. Zhang, et al., Distributed robotic systems in the edge-cloud continuum with ros 2: a review on novel architectures and technology readiness, in: FMEC, 2022.

[15] E. C. Ferrer, E. Jiménez, J. L. Lopez-Presa, J. Martín-Rueda, Following leaders in byzantine multirobot systems by using blockchain technology, IEEE Transactions on Robotics 38 (2) (2021) 1101–1117.

[16] S. R. Pokhrel, J. Choi, Federated learning with blockchain for autonomous vehicles: Analysis and design challenges, IEEE Transactions on Communications 68 (8) (2020) 4734–4746.

[17] Y. Lu, X. Huang, K. Zhang, S. Maharjan, Y. Zhang, Communication-efficient federated learning and permissioned blockchain for digital twin edge networks, IEEE Internet of Things Journal 8 (4) (2021) 2276–2288.

[18] A. Gueidi, H. Gharsellaoui, S. Ahmed, The rise of robotics data for real-time management based on new nosql solution, International Journal of Advanced Pervasive and Ubiquitous Computing (IJAPUC) 11 (2) (2019) 11–26.

[19] L. Asprino, E. Daga, A. Gangemi, P. Mulholland, Knowledge graph construction with a façade: a unified method to access heterogeneous data sources on the web, ACM Transactions on Internet Technology 23 (1) (2023) 1–31.

[20] E. Cuadros Zegarra, D. Barrios Aranibar, Y. Cardinale, Iort-based middleware for heterogeneous multi-robot systems, Journal of Sensor and Actuator Networks 13 (6) (2024) 87. doi:10.3390/jsan13060087.

[21] M. Imran, H. Yin, T. Chen, Q. V. H. Nguyen, A. Zhou, K. Zheng, Refrs: Resource-efficient federated recommender system for dynamic and diversified user preferences, ACM Transactions on Information Systems 41 (3) (2023) 1–30.

[22] N. Majcherczyk, N. Srishankar, C. Pinciroli, Flow-fl: Data-driven federated learning for spatio-temporal predictions in multi-robot systems, in: 2021 IEEE international conference on robotics and automation (ICRA), IEEE, 2021, pp. 8836–8842.

[23] Z. Chen, W. Xu, B. Wang, H. Yu, A blockchain-based preserving and sharing system for medical data privacy, Future Generation Computer Systems 124 (2021) 338–350.

[24] Z. Guan, X. Zhou, P. Liu, L. Wu, W. Yang, A blockchain based dual side privacy preserving multi party computation scheme for edge enabled smart grid, IEEE Internet of Things Journal (2021) 1–1.

[25] W. Wang, D. T. Hoang, P. Hu, Z. Xiong, D. Niyato, P. Wang, Y. Wen, D. I. Kim, A survey on consensus mechanisms and mining strategy management in blockchain networks, Ieee Access 7 (2019) 22328–22370.

[26] P. Abichandani, D. Lobo, S. Kabrawala, W. McIntyre, Secure communication for multiquadrotor networks using ethereum blockchain, IEEE Internet of Things Journal 8 (3) (2020) 1783–1796.

[27] F. Keramat, J. P. Queralta, T. Westerlund, Partition-tolerant and byzantine-tolerant decision making for distributed robotic systems with iota and ros2, IEEE Internet of Things Journal 10 (14) (2023) 12985–12998.

[28] T. T. Nguyen, A. Hatua, A. H. Sung, Blockchain approach to solve collective decision making problems for swarm robotics, in: Congress on Blockchain and Applications, 2019, pp. 1–12.
[29] A. Kapitonov, S. Lonshakov, A. Krupenkin, I. Berman, Blockchain-based protocol of autonomous business activity for multi-agent systems consisting of uavs, in: 2017 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED-UAS), 2017, pp. 84–89.
[30] J. Li, J. Wu, J. Li, A. K. Bashir, M. J. Piran, A. Anjum, Blockchain-based trust edge knowledge inference of multi-robot systems for collaborative tasks, IEEE Communications Magazine 59 (7) (2021) 94–100.
[31] S. Salimi, J. P. Queralta, T. Westerlund, Hyperledger fabric blockchain and ros 2 integration for autonomous mobile robots, in: 2023 IEEE/SICE International Symposium on System Integration (SII), IEEE, 2023, pp. 1–8.
[32] J. Grey, I. Godage, O. Seneviratne, Blockchain-based mechanism for robotic cooperation through incentives: Prototype application in warehouse automation, in: Proceedings of the 2021 IEEE Blockchain Conference, IEEE, 2021.
[33] Y. Lu, X. Huang, Y. Dai, S. Maharjan, Y. Zhang, Blockchain and federated learning for privacy-preserved data sharing in industrial iot, IEEE Transactions on Industrial Informatics 16 (6) (2020) 4177–4186.
[34] H. Chai, S. Leng, Y. Chen, K. Zhang, A hierarchical blockchain-enabled federated learning algorithm for knowledge sharing in internet of vehicles, IEEE Transactions on Intelligent Transportation Systems 22 (7) (2021) 3975–3986.
[35] Y. Xianjia, J. P. Queralta, J. Heikkonen, et al., Federated learning in robotic and autonomous systems, Procedia Computer Science 191 (2021) 135–142.
[36] J. Zhang, T. Xie, Y. Jing, et al., Bora: a bag optimizer for robotic analysis, in: SC20: International Conference for High Performance Computing, Networking, Storage and Analysis, 2020, pp. 1–15.
[37] A. Yang, S. Zhu, X. Li, J. Yu, M. Wei, C. Li, The research of policy big data retrieval and analysis based on elastic search, in: 2018 International Conference on Artificial Intelligence and Big Data (ICAIBD), IEEE, 2018, pp. 43–46.
[38] X. Yu, L. Li, X. He, S. Chen, L. Jiang, Federated learning optimization algorithm for automatic weight optimal, Computational Intelligence and Neuroscience 2022 (1) (2022) 8342638.
[39] L. Su, J. Xu, P. Yang, A non-parametric view of fedavg and fedprox: Beyond stationary points, Journal of Machine Learning Research 24 (203) (2023) 1–48.
[40] J. Wang, et al., A novel framework for the analysis and design of heterogeneous federated learning, IEEE Transactions on Signal Processing 69 (2021) 5234–5249.
[41] H. B. McMahan, et al., Communication-efficient learning of deep networks from decentralized data, in: International Conference on Learning Representations (ICLR), 2023.
[42] G. Zhang, F. Pan, Y. Mao, S. Tijanic, M. Dang'Ana, S. Motepalli, S. Zhang, H.-A. Jacobsen, Reaching consensus in the byzantine empire: A comprehensive review of bft consensus algorithms, ACM Computing Surveys 56 (5) (2024) 1–41.
[43] Autorobo, https://www.nooploop.com/en/autorobo/, accessed on 2023-07-21.
[44] L. Pichierri, A. Testa, G. Notarstefano, Crazychoir: Flying swarms of crazyflie quadrotors in ros 2, IEEE Robotics and Automation Letters.
[45] Nokov, https://en.nokov.com/direct, accessed on 2023-07-21.
[46] A. Makris, K. Tserpes, G. Spiliopoulos, D. Zissis, D. Anagnostopoulos, Mongodb vs postgresql: A comparative study on performance aspects, GeoInformatica 25 (2021) 243–268.
[47] H. Matallah, G. Belalem, K. Bouamrane, Comparative study between the mysql relational database and the mongodb nosql database, International Journal of Software Science and Computational Intelligence (IJSSCI) 13 (3) (2021) 38–63.
[48] C. Xu, R. Su, R. Wang, S. Duan, Self-attention factor graph neural network for multiagent collaborative target tracking, IEEE Internet of Things Journal 11 (20) (2024) 32381–32392. doi:10.1109/JIOT.2024.3370830.
[49] Y. Romano, A. Aberdam, J. Sulam, M. Elad, Adversarial noise attacks of deep learning architectures: Stability analysis via sparse-modeled signals, Journal of Mathematical Imaging and Vision 62 (2020) 313–327.
[50] M. Baruch, G. Baruch, Y. Goldberg, A little is enough: Circumventing defenses for distributed learning, in: Advances in Neural Information Processing Systems, Vol. 32, 2019.
[51] C. Xie, O. Koyejo, I. Gupta, Fall of empires: Breaking byzantine-tolerant sgd by inner product manipulation, in: Uncertainty in Artificial Intelligence, PMLR, 2020, pp. 261–270.
[52] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, J. Stainer, Machine learning with adversaries: byzantine tolerant gradient descent, International Conference on Neural Information Processing Systems (2017) 118–128.
[53] S. P. Karimireddy, L. He, M. Jaggi, Learning from history for byzantine robust optimization, International Conference on Machine Learning (2021) 5311–5319.
[54] Y. Chen, L. Su, J. Xu, Distributed statistical machine learning in adversarial settings: Byzantine gradient descent, Proceedings of the ACM on Measurement and Analysis of Computing Systems 1 (2) (2017) 1–25.
[55] D. Yin, Y. Chen, R. Kannan, P. Bartlett, Byzantine-robust distributed learning: Towards optimal statistical rates, International Conference on Machine Learning (2018) 5650–5659.

Ran Wang received the B.E. degree from the Beijing Information Science and Technology University, China in 2013, and the M.S. degree from the University of Science and Technology Beijing (USTB), China in 2016. She is currently working toward the Doctoral degree in the Microarchitecture and Integrated Circuits Lab (MICL) at University of Science and Technology Beijing. Her research interests include multi-agent systems, distributed security and internet of things.

Sisui Tang is currently working toward the Master degree at University of Science and Technology Beijing. Her research interests include distributed security, confidential computing and internet of things.

Hangning Zhang is currently working toward the Master degree at University of Science and Technology Beijing. Her research interests include distributed security, confidential computing and internet of things.

Shihong Duan received Ph.D. degree in computer science from University of Science and Technology Beijing (USTB). She is an associate professor with the School of Computer and Communication Engineering, USTB. Her research interests include wireless indoor positioning, multi-robots network and internet of things.

Xiaotong Zhang received the M.S., and Ph.D. degrees from University of Science and Technology Beijing, in 1997, and 2000, respectively. He is a Professor in the Department of Computer Science and Technology, University of Science and Technology Beijing. His research includes wireless sensor networks, networks management, signal processing of communication and computer architecture.

Cheng Xu received the B.E., M.S. and Ph.D. degree from the University of Science and Technology Beijing (USTB), China in 2012, 2015 and 2019 respectively. He is currently working as an Associate Professor in the Microarchitecture and Integrated Circuits Lab (MICL) at University of Science and Technology Beijing. He was supported by the Post-doctoral Innovative Talent Support Program from Chinese government in 2019. He is an associate editor of International Journal of Wireless Information Networks. His research interests now include swarm intelligence, multi-robots network, distributed security and internet of things. He is a member of the IEEE.