AuthROS: Secure Data Sharing Among Robot Operating Systems Based on Ethereum

Shenhui Zhang*, Wenkai Li*, Xiaoqi Li*‡, Boyi Liu†‡
*School of Computer Science and Technology & School of Cyberspace Security, Hainan University, Haikou, China
†Robotics Institute, The Hong Kong University of Science and Technology, Hong Kong SAR, China
Email: csxqli@gmail.com, by.liu@ieee.org

***Abstract*—The Robot Operating System (ROS) streamlines human processes, increasing the efficiency of various production tasks. However, the security of data transfer operations in ROS is still in its immaturity. Securing data exchange between several robots is a significant problem. This paper proposes *AuthROS*, an Ethereum blockchain-based secure data sharing method, for robot communication. It is a ROS node authorization system capable of ensuring the immutability and security of private data flow between ROS nodes of any size. To ensure data security, AuthROS employs the smart contract for permission granting and identification, SM2-based key exchange, and SM4-based plaintext encryption techniques. In addition, we deploy a data digest upload technique to optimize data query and upload performance. Finally, the experimental findings reveal that AuthROS has strong security, time performance, and node forging in cases where data should be recorded and robots need to remain immobile.**

***Keywords*—ROS; Blockchain; Encryption Algorithm**

I. Introduction

The Robot Operating System (ROS) [1] is an open-source meta-operating system for robots. It is a distributed multi-process framework based on message communication. ROS is designed to improve the code reuse rate in the field of robotics research and development. ROS provides functions similar to those provided by operating systems (OS), such as hardware abstraction description, low-level driver management, etc. The essence of ROS is a TCP/IP-based Socket communication mechanism [2], which is capable of performing several types of communication, such as service-based synchronous RPC communication, topic-based asynchronous data stream communication, and parameter server-based data storage. This flexible framework enables different modules of ROS to be designed separately and loosely coupled at runtime.

However, there are some obvious drawbacks to ROS [3]–[5]. First of all, ROS lacks measures to ensure data security. Attackers can steal data from ROS utilizing the Publisher-Subscriber mode, which endangers the immobility and safety of data. In addition, ROS also has problems with data reliability, and data passed among ROS nodes based on this framework can be intercepted or forged [6]. For example, these defects in autonomous driving would result in a serious accident. These two defects lead to insecure and unstable data exchange and sharing in the scenario of multi-robot interaction based on ROS.

Furthermore, the issues raised by ROS had been alleviated with the assistance of SROS [7], a typical scheme based on Transport Layer Security (TLS) and certificate mechanisms. In the ROS 2.0 phase, integrating Data Distribution Service (DDS) components and SROS enhances the scheme's authorization and access control security. ROS 2.0 implements centralized access control utilizing policies including permission control and certificate signing. The DDS security standard indicators [8] include Authentication, Access Control, Cryptographic, Logging, and Data Tagging. The ROS 2.0 have not achieved Logging and Data Tagging, specifically, the capacity to record data and behaviour. As a result, when an attacker gains central control, the data can be manipulated directly, and there is no way to trace back error information.

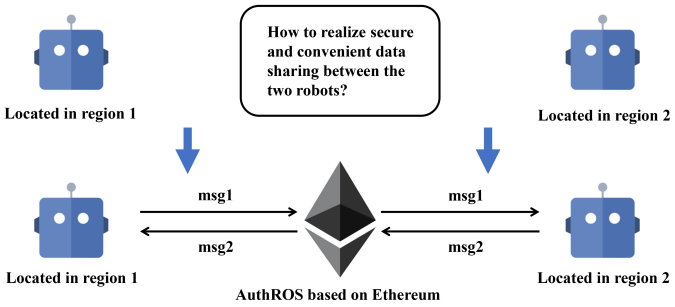


Figure 1. The design idea of the AuthROS framework. To reduce the risk brought by attacks during data sharing in ROS, Ethereum is used to design this data sharing framework.

Therefore, we leverage Ethereum blockchain [10] to solve the above drawbacks and propose a novel framework named AuthROS (Authority in Robot Operating Systems), whose design idea is shown in Fig. 1. In order to ensure the security of the entire communication network and data, AuthROS leverages a series of remarkable cryptographic algorithms, i.e., the SM algorithm family [11]. At present, three types of algorithms are mainly applied: SM2, SM3, and SM4 [11]. Based on the SM algorithms and blockchain technology, we can finally solve the existing defects of ROS. AuthROS achieves an efficient and secure data sharing framework for ROS based on blockchain technology, Web3, and SM algorithms. And due to the data traceability of blockchain systems, data logging for monitoring abnormal node behavior can be achieved. It allows critical data captured by robots shared by other robots with authority in the same Ethereum network.

At the same time, the confidential data is also stored in the Ethereum network for later check. Furthermore, it has virtually no restrictions on the types of data, most types of messages are supported to interact with the Ethereum network. The algorithm encryption system based on the SM algorithm

‡ The corresponding authors.

family can effectively ensure the security of data transmission.

The main contributions of this paper are as follows:

(1) We propose a novel framework called AuthROS based on blockchain technology. To the best of our knowledge, it is the *first* secure data-sharing framework for robots loaded with ROS.
(2) AuthROS has a functionality of authority granting controlling access to specified confidential data transmitted among ROS. Furthermore, it can conduct secure encrypted communication leveraging SM algorithms to prevent attacks (e.g., Node Forging).
(3) AuthROS achieves a secure, reliable, and convenient interaction solution for ROS-based robots leveraging the Ethereum blockchain. Evaluation of the process for generating digest from 800KB encrypted data reveals that AuthROS is efficient, completing in 6.34ms.

The rest of this paper is organized as follows. Section 2 introduces the background and Section 3 summarize the related work. Section 4 introduces the framework of AuthROS. Then we analyzes the results of evaluation experiments in Section 5 and finally conclude the paper in Section 6.

II. Background

In this section, we introduce ROS and Ethereum, since AuthROS consists of the ROS system and the Ethereum network.

A. ROS.

ROS is a distributed process framework to promote the high reusability of robotic software systems. It is an open-source, meta-operating system that provides adaptable and practical qualities for robot manipulations [9]. Some abstractions in ROS developed from the OS offer services comparable to the operating system, including standard hardware APIs, low-level device management, message transmission between nodes, and package management for application distribution. ROS also includes a Peer-to-Peer (P2P) network topology that blends service-based synchronous Remote Procedure Call (RPC) communication, topic-based asynchronous data flow communication, and others.

Terms. Each of the software modules is a node [1]. And the nodes communicate with each other by passing messages, which are strongly typed and support multiple nesting. Another "odometry" or "map" type term is "topic," which refers to a way of communication in nodes, from which numerous publishers and subscribers complete the message transmission. Finally, a service consists of a string name, a request message, and a response message. However, the network communication protocol it uses does not handle synchronous transactions.

B. Ethereum.

Ethereum [10], a popular blockchain platform, is another helpful technology. Blockchain technology aims to record all transactions in the network to safeguard data. It generates users' addresses using elliptic curve algorithms and hashing algorithms before authenticating transactions.

Geth. Geth is an Ethereum client built in Golang language, and the local machine can join the Ethereum P2P network as a node after the running of Geth [26]. In this paper, the Geth is used to build an Ethereum private network. Ethereum supports Externally Owned Accounts (EOA) and smart contract accounts. With the exception of the network administrator, who has a contract account, all AuthROS robots are associated with EOA accounts. The first 20 bytes of the SHA3 hash of a user's public key serve as the account's index [27].

Consensus Algorithms. Consensus algorithms provide the immutability, automation, and anonymity of blockchain transactions. Consensus algorithms maintain the meaning and value of blockchain technology as a distributed database. It ensures that the states of the blocks on the chain remain consistent. Proof of Work (PoW) [27], Proof of Authority (PoA) [28] are the consensus methods for Ethereum in AuthROS. Nodes in networks using PoW and PoA consensus algorithms have different roles as miners or validators. PoW relies on mining operations to validate blocks, whereas PoA employs trusted nodes that are pre-authorized.

Smart Contract. The smart contract is an executable software program that can be interacted with peers on the network [29]. It has increased the scalability of blockchain. Users can execute customized transaction rules in smart contracts, and transactions are irreversible once completed. Additionally, smart contracts can be programmed in a Turing complete language known as Solidity, Vyper, etc. Peer-based decision-making is enabled through carefully built smart contracts in applications such as IoT, multi-robot systems, and smart cities.

III. Related Work

Large-scale applications of robots will inevitably involve many problems, such as the security of data transmission, data sharing, and the classical Byzantine problem [12]. With the development of ROS 2.0, several studies have concentrated on providing powerful tools for secure robots interactions, and pursuing complete DDS standard indicators [8].

Sundaresan et al. [13] proposed an access control strategy based on IPSec to solve the problem of identity authentication and encrypted robot communication. It ensured that IP packets were encrypted and authenticated by modifying the master node and client libraries. Nonetheless, it restrictd access due to the control of user permissions through IP.

Ruffin et al. [7] proposed SROS, which was based on Transport Layer Security (TLS) and certificate authentication mechanisms to achieve identity authentication, encryption, and access control of communication. Thus, ports are assigned to robots at runtime. Multiple robots can access them simultaneously, with centralized access control ensured by the security protocol and identity authentication mechanism.

Combining Datagram TLS and TLS, Breiling et al. [14] proposes a secure channel for node-to-node communication. It requests the initial handshake using the certificate and RSA encryption, which is encrypted using the AES algorithm. And it utilizes Message Authentication Codes (MACs) following data transmission to ensure data integrity.

However, none of the above work has solved the problem of data traceability. Due to the information record function and security performance of blockchain, it has been favored by researchers in the robot community, and much research on the integration of robots and blockchain has been carried out.

Some research focusing on swarm communication is listed as follows. To combat COVID-19 and break through the bottleneck of existing multi-swarming UAVs based on 5G, Rajesh Gupta et al. [15] proposed a blockchain-envisioned software multi-swarming UAV communication scheme based on a 6G network with intelligent connectivity. Pranav K. Singh et al. [16] proposed an efficient communication framework for swarm robotics based on PoA consensus to break through the limitations of existing robotic control and communication schemes. Eduardo Castelló Ferrer et al. [17] introduced the first learning framework for secure, decentralized, and computationally efficient data and model sharing among multiple robot units installed at multiple sites. Pramod et al. [18] used a set of experiments to validate that Ethereum can be a secure media for communication for multiple small Unmanned Aerial Vehicles (sUAVs).

Research focus on secure information sharing also weighs a lot. Alsamhi et al. [19] proposed a framework to facilitate information sharing within multi-robot using Ethereum. This framework proved to be effective. Nishida et al. [20] introduced a methodology to share information among autonomous robots and demonstrated through experiments how the differences in data size recorded in the blockchain affect the chain size. Jorge Peña Queralta et al. [21] presented a novel approach to managing collaboration terms in heterogeneous multi-robot systems with blockchain. This approach can estimate the available computational resources of different robots and integrate information about the environment from different robots, to evaluate and rank the quality and accuracy of each of the robots' sensor data. Vasco Lopes et al. [22] proposed an architecture that uses blockchain as a ledger and smart contract for robotic control by using external parties, Oracles, to process data. The proposed architecture shows great potential for secure information sharing between robots.

There comes the classic Byzantine problem with Robotic swarms. In the survey of Eduardo Castelló Ferrer et al. [23], a set of Byzantine Follow The Leader (BFTL) problems were presented, and algorithms to tackle the BFTL problems based on blockchain were proposed too. Alexandre Pacheco et al. [24] presented a robot swarm composed of Pi-puck robots that maintain a blockchain network. The blockchain served as a security layer to neutralize Byzantine robots. Volker Strobel et al. [25] demonstrated how robotic swarms achieve consensus in the presence of Byzantine robots exploiting blockchain technology.

IV. Framework and Elements of AuthROS

In this section, we explain the AuthROS framework sharing data based on the blockchain, the SM algorithms, and ROS. AuthROS leverages encryption technology, consensus mechanisms, and smart contract to assure security in the data generation, transmission, and sharing process.

A. Assumptions.

To ensure the availability and efficiency of AuthROS, we incorporate some assumptions into its design. The role of AuthROS will likewise be heavily influenced by these assumptions. The following assumptions are made:

Blockchain Security. Due to the features of distribution and encryption in blockchain, it possesses superior security performance. Therefore, we assume that Ethereum, as a channel for sharing information, is secure and trustworthy.

Robots Manager. We presume that any robot or robot cluster is capable of maintaining by more than one manager for data communication. These managers refer to the Core Users of AuthROS (CURA), which serves as data sharing.

Identity Knowability. Different CURAs only employ AuthROS for data sharing after establishing a trusting connection. In other words, the foundation of data exchange is that the CURAs are confident with each other. Any CURA cannot reveal the EOA address that confirms its identification to a third party.

Unique Means of Sharing. CURAs will only exchange data via AuthROS. It is essential for the accessibility of AuthROS.

Administrator. There is always an administrator on the Ethereum private network who handles membership additions and other emergencies.

Due to the immutability, semi-decentralization, and anonymity of the private chain, it is ideally suited as a platform for information sharing in AuthROS. The blockchain network and hardware platform will then be described in depth.

B. Blockchain Network.

We select PoA and PoW as the consensus algorithms for AuthROS on Ethereum. The characteristics of blockchain networks based on two distinct consensus algorithms are as follows:

Same Contract. No matter which consensus mechanism the network is employed, we all utilize the same contract for consistent internal interfaces.

Same Block Difficulty and Quantity of Users. To examine the applicability of the consensus technique in the subsequent tests, we not only simulate the same number of users in the blockchain network based on the two consensuses, but also we set the block difficulty to the same value in the genesis block.

Ethereum is utilized to build the blockchain network of AuthROS. The private network of AuthROS is developed based on Geth. The smart contract is deployed with EVM (Ethereum Virtual Machine). PoW and PoA are two different consensus algorithms, and we evaluate the effects of the two on the time performance of AuthROS. Furthermore, to address the limited computing resources in smart terminal devices, we build a private Ethereum network in the cloud server. The robots can obtain the connection to the blockchain of AuthROS

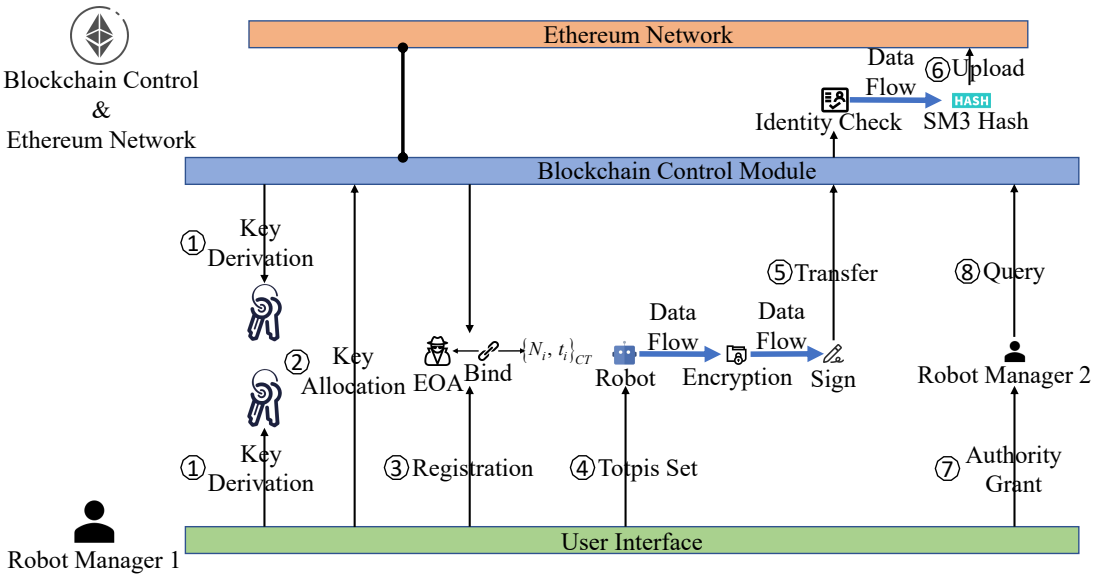


Figure 2. The framework of AuthROS. The User Interface, Blockchain Control Module, and Ethereum Network are represented by the green, blue, and orange boxes, respectively. And the blue and black arrows are the flow of data and operations, respectively.

by the local ROS master running on the host with access to the internet.

Due to the robot's limited processing power, building an Ethereum node there will encounter the computational bottleneck typical of Edge Computing. Therefore, as Fig. 3 depicted, the Ethereum network composed of 3 nodes is implemented using a host. The sole responsibility of the robot is to maintain contact with the host's ROS Master. If ROS is kept isolated from the Ethereum network, the Edge Computing bottleneck can be ignored. The miner node serves as the bootstrap node and is in charge of bringing together other nodes to create the overlay network. Moreover, to interact with the Ethereum network, four ROS-Melodic robots are connected to two EOA accounts in Node2 and Node3, respectively.

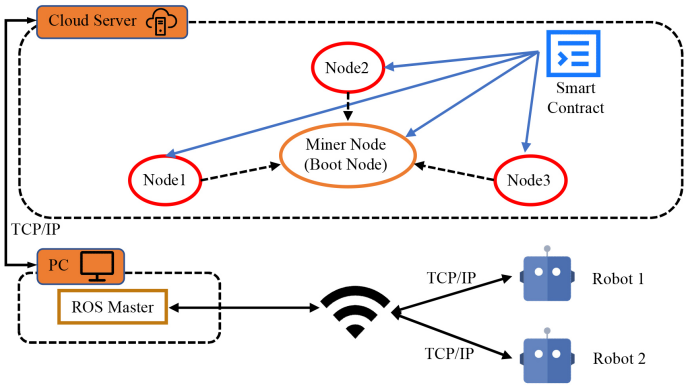


Figure 3. The framework of the private Ethereum network of AuthROS. The blockchain network is deployed in the cloud server and contains three nodes. Miner Node is the boot node, which connects other nodes, such as Node1 and Node2. All robots connect to the blockchain network through a node within the ROS master in local ROS.

C. AuthROS Framework and Process

For the process of AuthROS in Fig. 2, users should first upload their name/token key-value pairs. And an SM4 key and an SM2 public key are required to authenticate the digital signature, which is necessary for the Identity Check and Authority Grant. Users can choose a topic to monitor after registering the identification. The monitored topic often forwards some essential information, such as the data in radar, camera, and other sensors. As soon as the topic publishes data, AuthROS will immediately launch a subscriber to capture and parse the topic's contents. The data will be delivered to the network management module of AuthROS after being encrypted by SM4 and signed by SM2. After the validation of data ciphertext by the SM2 signature, the SM3 hash method generates the data digest. The value of the digest will be posted to the blockchain network. Users can grant access to their shared data to other users on the chain. The user's identity is represented by a unique Ethereum external account (EOA) in the Ethereum network, and authority is granted primarily via the exchange of SM4 keys uploaded by the user.

This framework possesses the following characteristics:

Plasticity. The AuthROS uses a private chain, which is more flexible in terms of block time and consensus conditions. And the semi-decentralized structure of the private chain makes it add new members to the network more conveniently.

Process Security. The AuthROS uses SM4, a symmetric encryption technique, to encrypt all data transmission and interaction operations. The data digest interaction system combined with Identity Check and Authority Grant mechanism can ensure data integrity, security, and immutability.

D. Data Sharing Protocol

In this section, we will introduce the core mechanism in AuthROS, including the key distribution protocol, data

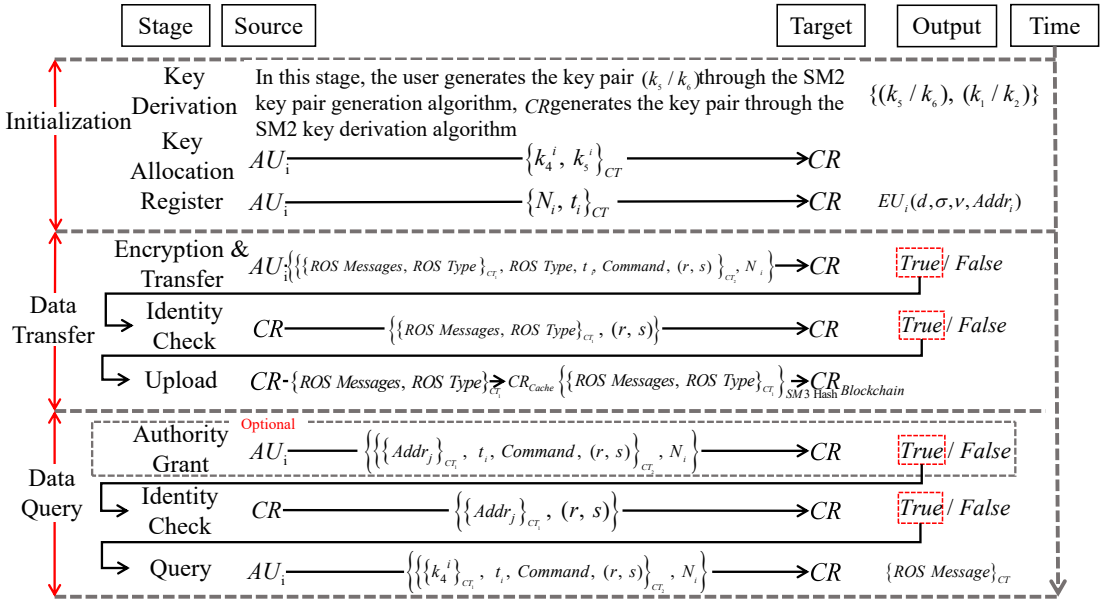


Figure 4. The process of data sharing in AuthROS. The whole process is worked from top to bottom, and each dotted box is a phase.

encryption scheme, etc. Due to the possibility of data being intercepted and altered during transmission, there are some stricter standards for data integrity and security in AuthROS. Consequently, AuthROS developed an interaction strategy based on data digests that uses the SM2 digital signature algorithm and the SM3 hash algorithm. In this strategy, the server locally maintains all of the data while the blockchain network uploads the data digest. The notations used in the design are summarized in Table I.

The design of AuthROS consists of three phases: (1) Initialization, (2) Data Transfer, and (3) Data Query. Fig. 4 depicts the subprocess corresponding to each level.

E. Key Generation.

Throughout the whole data sharing life cycle, we must verify its validity, integrity, and non-repudiation. We thus generate a pair of asymmetric keys for each user and sign their shared data. To satisfy the aforementioned conditions, AuthROS implements the production and verification of the elliptic curve public key by referencing the key pair generation and public key verification criteria introduced by the SM2 algorithm [31].

Assuming that the private key is d_C^i, and AuthROS computes the public key $P_C^i = \left[d_C^i G\right]$ using the multiple points fast algorithm of multiple elliptic curves [30], where G is the base point of the elliptic curve and its order is a prime number. In the meantime, AuthROS also employ the SM2 key derivation function to build the appropriate public/private key pair $\left(P_S^i, d_S^i\right)$ for key exchanges. Specific algorithms can be found in the standard for SM2 algorithms [31].

F. Key Allocation.

Data is often transferred in plaintext across the transmission connection in the current TCP/IP network transmission framework, making it available for malevolent users of a third party to intercept the data and conduct a series of assaults such as replay attacks and man-in-the-middle attacks. To maintain the security of data during transmission and storage on the blockchain, it is necessary to encrypt and authenticate the data using cryptographic technology. AuthROS implements the key distribution function to ensure secure storage of blockchain with SM4 and SM2. Among them, the SM4 key encrypts plaintext data and the SM2 public key confirms the digital signature.

Firstly, the user enters their own SM4 key K_C^i and SM2 public key P_C^i. Then, the user selects a system public key $P_S^i \in k_s$ to encrypt the message $SM2E : \left\{\left(K_C^i, P_C^i\right), P_S^i\right\} \rightarrow CT$, where d_S^i and $P_S^i \in k_s$ form a public/private key pair $\left(P_S^i, d_S^i\right)$, k_s is a set of SM2 public keys regularly published by us. In addition, user should use their personal SM2 private key d_C^i to sign the resulting ciphertext $CT, SM2S : \left\{CT, d_S^i\right\} \rightarrow (r, s)$. Then, adding an information frame after the ciphertext CT to produce the final message $\left\{CT, (r, s), P_S^i, Command\right\}$, where $Command$ denotes the instructions of operations will be processed. The message is sent to the CR via the TCP/IP protocol stack. CR will decode the ciphertext with the SM2 private key d_S^i corresponding to P_S^i, $SM2D : \left\{CT, d_S^i\right\} \rightarrow \left(K_C^i, P_C{}^i\right)$ after receiving the data packet. After decryption, CR will utilize P_C^i to validate the ciphertext's authenticity and integrity using CT, $SM2V : \left\{CT, P_C^i, (r, s)\right\} \rightarrow True\ /\ False$. After the integrity and accuracy checks, the keys K_C^i and P_C^i will be stored in the blockchain for next operations.

1) Registration.: At this procedure, user AU_i registers with AuthROS and uploads a key-value pair (N_i, t_i) as the identity. Then, the N_i supplied by the user and t_i, the SM4 key K_C^i uploaded by the user during the key allocation, the

TABLE I
SUMMARY OF NOTATIONS IN THIS PAPER.

Notation	Meaning
AU_i	The i_{th} user of AuthROS
$Type$	Data type of ROS topic transmission
$LV_{x,y,z}$	Three axis velocity of data of odometry
$AV_{x,y,z}$	Triaxial angular velocity of odometry
$Pose$	Pose information contained in data of odometry
Cov	Covariance information contained in data of odometry
TS	Timestamp information contained in odometry
T	Time of ROS data captured
CR	Cloud server of AuthROS deploying Ethereum
ND	Plaintext data
CT	Ciphertext of plaintext
$Addr_i$	Address of the identity of the i_{th} user
d, σ, v	Structure for users to store shared data, their own keys, and keys of other users
$EU_i\,(d, \sigma, v, Addr_i)$	Corresponding identity of the i_{th} user
N/d	Username/password pair
k_s	A set of SM2 public keys published regularly
$\left(P_S^i, d_S^i\right)$	The SM2 public/private key pairs published regularly in the system
K_C^i	User's SM4 key
$\left(P_C^i, d_C^i\right)$	Public/private key pair for signing and verifying
(r, s)	SM2 signature value
$SM2E : \left\{ND, P_S^i\right\} \rightarrow CT$	The process of generating ciphertext CT by encrypting plaintext data ND with SM2 public key P_S^i
$SM2D : \left\{CT, P_S^i\right\} \rightarrow ND$	The process of decrypting CT using d_S^i to get ND
$SM4 : \{ND, K_C\} \rightarrow CT$	The process of generating data ciphertext CT by encrypting plaintext data ND with SM4 key K_C
$SM2S : \left\{Mradw, d_S^i\right\} \rightarrow (r, s)$	The data to be sent is signed with SM2 algorithm

SM2 public key P_C^i, and an unused account address $Addr_i$ on the Ethereum network are used to generate a mapping $N_i \rightarrow \left(t_i, K_C^i, P_C^i, Addr_i\right)$, In the preceding phase of key allocation, the user-uploaded SM4 key K_C^i will now be assigned to $\sigma, \sigma = K_C^i, v$ and v will remain empty, which means $(v = Null) \wedge (d = Null)$. The user's identification in the blockchain $EU_i\,(d, \sigma, v, Addr_i)$ will be created.

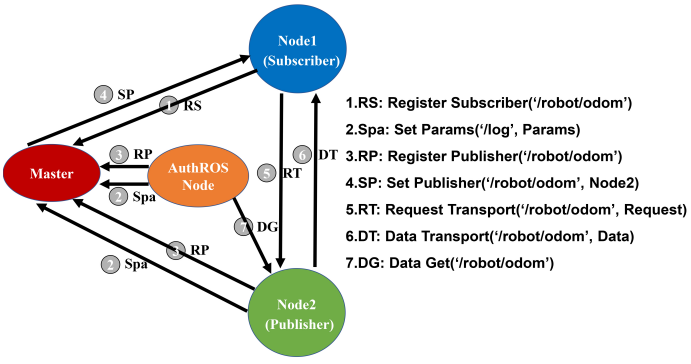


Figure 5. The workflow of ROS in local robots. Firstly, node1, node2, and AuthROS node should register their identity (subscriber or publisher) through ROS Master and set the topic '/robot/odom' for message transmission. Then, the AuthROS node can seize data at any given time when the communication between node1 and node2 is going on. Then, extracted messages will go through a range of format conversion and encryption.

G. Topic Set.

The core of AuthROS is a monitoring node. AuthROS is capable of monitoring the topics they want by inputting the names and message types of topics by users. Corresponding processing operations will be made for different message types. For example, for odometry-type messages, the AuthROS parses the respective messages after obtaining them from a topic named '1/robot/odom', and extracts useful information like three-axis velocity and angular velocity at a given time, etc. Fig. 5 shows how AuthROS obtain and parse the odometry-type messages.

1) Encryption and Transfer.: We use the SM4 encryption method [32] to encrypt the ND to get the ciphertext CT. CT is packaged with some necessary information frames and sent to CR. The odometry-type data in ROS consists of information such as $LV_{x,y,z}, AV_{x,y,z}$. Before transmission, the data must be packaged into $\{LV_{x,y,z}, AV_{x,y,z}, Pose, TS, Cov, Type\}_{ND_1}$ and encrypted with SM4 for the first time $SM4 : \left\{ND_1, K_C^i\right\} \rightarrow CT_1$. The user signs the CT_1 with the SM2 private key d_C^i to generate the signature value $SM2S : \left\{CT_1, d_S^i\right\} \rightarrow (r, s)$, encapsulates the information frame $\{CT_1, Type, t_i, T, Command, (r, s)\}_{ND_2}$ for the received ciphertext CT_1, re-encrypts the $ND_2, SM4 : \{ND_2, K_C\} \rightarrow CT_2$, encapsulates the information frame $\{CT_2, N_i\}_{ND_3}$ for the CT_2, and transmits it to the CR. After the identity check is successful, AuthROS will decrypt CT_2 to CT_1 and follow

up the ciphertext CT_1 with the data digest interaction scheme according to $Command$. However, for non-general data types, such as image data, matrix and compression are necessary as pre-processing steps.

V. Implementation and Evaluation

This section introduces the hardware platform and smart contracts in experiments and then analyzes AuthROS of response time from different perspectives: Consensus Algorithms, Message Size, and Efficiency of the SM algorithm family. We evaluate the response time of data upload based on 4 ROS-Melodic robots with a ROS Master and a private Ethereum network on the host. For the same configuration of robots, we only evaluate the performance of a single robot.

A. Hardware Platform Equipment

The robots we used to equip with a Jetson Nano B01 (Quad-core ARM A57 64-bit @1.43Ghz 4GB LPDDR4-3200), a controller which has a built-in 9-axis IMU sensor, RPLIDAR A1 radar, and a Wi-Fi module that can provide up to 867mbps communication bandwidth and an RGB-D binocular camera. To realize the autonomous movement of the robot in the closed experimental environment, Visual Slam (Visual Simultaneous Localization and Mapping) and Lidar Slam (Lidar Simultaneous Localization and Mapping) are combined to build a complete map of the closed experimental space. ROS Melodic is set up in every robot, as long as the personal computer running ROS Master. Through the Wi-Fi module, each robot can connect to the ROS Master to achieve stable communication. The PC running ROS Master also connects to the server hosting the Ethereum network, thus can provide interaction between robots and Ethereum.

The controller and Jetson Nano are connected through a UART connection using software function calls provided by the controller's onboard C++ SDK. The controller collects the 9-axis IMU sensor and motor data. The RGB-D and RPLIDAR are connected to the Jetnano to capture images and collect lidar data. Motion commands are communicated between the Jetson Nano and controller to realize motion planning and control. The cloud server hosting Ethereum Network with a CPU (Intel Xeon (Ice Lake) Platinum 8369B @3.5GHz), memory (16GB DDR4 3200MHz), and disk (80GB ESSD).

B. Ethereum Smart Contract Implementation

The smart contracts developed in experiments are written in Solidity v7.6. The contracts allowed for identity registration, knowledge-upload, authority-grant, etc. All functions are listed as follows:

Register(bytes). This function registers an identity in the Ethereum network inputting a parameter of Bytes-type. The SM4 key is used for data encryption as a token. The robot owner converted the SM4 key to Bytes-type.

Data Upload(bytes, bytes, bytes). The robot owner uploads confidential data to the Ethereum network for immutable persistent storage by calling this method. This function accepts three parameters of Bytes-type, the first parameter is the data ciphertext of Bytes-type to be uploaded. The second parameter is a Bytes-type token (SM4 key) that indicates the identity of the data uploader. The third parameter is the timestamp of Bytes-type when this method was called.

Authority Grant(address). The robot that calls this method will append the token (SM4 key) that indicates its identity to the token list in the specified EOA account so that the specified account will have access to the function caller's data. This function accepts a parameter of address-type, which is the address of the EOA that will be given access to the function caller's data.

Data Query(bytes, address). The user can call this function to query the data being shared by the target EOA, when the user's token exists in the target EOA's token list. The first parameter "bytes" is a Bytes-type token that indicates the identity of the function caller, and the second parameter "address" indicates the target EOA address for the query operation.

C. Evaluation Process

We take secure image sharing as an example in the AuthROS experiment. In the process of crime scene investigation and evidence collection, the images taken by different robots at the crime scene have strict requirements on security performance. User authorization is required for archiving and retrieval. We will carry out experiments against this, and the process of which is depicted in Fig. 6.

Firstly, we start the robot and load the ROS master on the PC. After the initialization is complete, the robot will automatically connect to the ROS master according to the genesis block. We get the facial data we need from the topic '/robot/CompressedImage' in the form of a picture of 58 KB, and then use the OpenCV toolkit to matrix it. Next, the matrix will be converted into a character string and encrypted by SM4 algorithms. The ciphertext will be signed with the user's SM2 public key and transmitted to the cache loaded in the server of AuthROS. At the same time, the SM3 cipher hash algorithm is used to generate the abstract of ciphertext, and the smart contract sends a transaction with the abstract to the Ethereum network with the help of the interface Data_Upload.

Finally, the authorized robot owner can invoke the interface Data_Check to check the ciphertext. The idea of homomorphic encryption is used for reference to design the process of data checking. The ciphertext within the cache is hashed by SM3, and the generated hash value is compared with the one queried from Ethereum. To ensure the immutability of data, if the two abstracts are the same, the Redis, an in-memory storage structure, will return ciphertext to the user, otherwise, return an error. At the same time, once a robot is authorized, it means that its owner has obtained the SM4 key of the data sharer. After the ciphertext is queried, the corresponding SM4 key can be used to decrypt the ciphertext, and the OpenCV can also be used to restore the image. The whole process can be seen in the video.

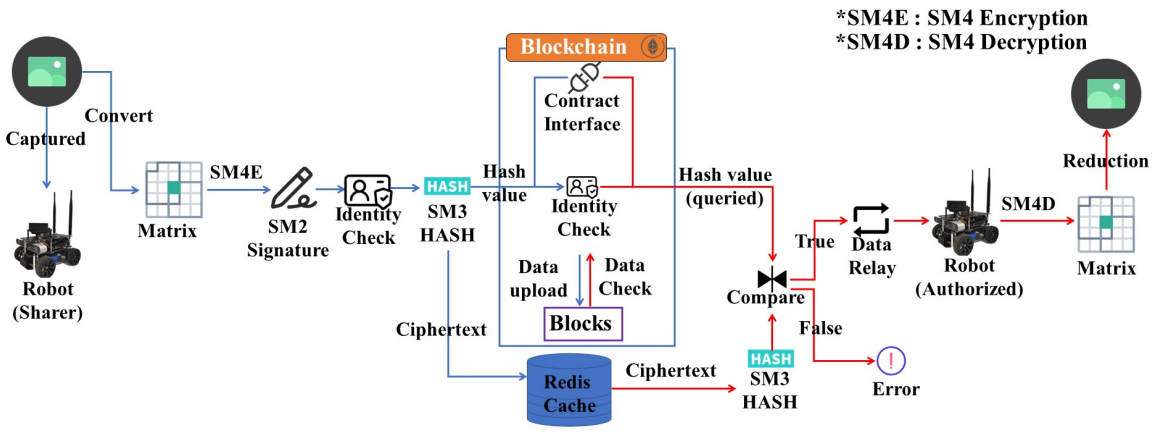


Figure 6. Process of image sharing. Images captured by robots are converted into a matrix. Then the matrix will be encrypted by the SM4 encryption algorithm, signed by the SM2 signature algorithm, and transmitted to the cloud server for persistence storage. The image is queried using the SM3 hash and ciphertext check.

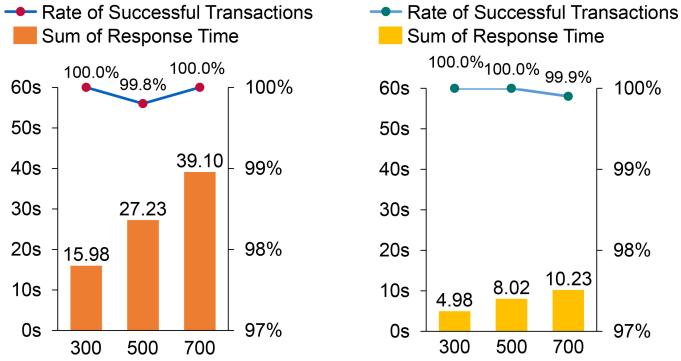


Figure 7. Left: total response time and success rate of PoW. Right: total response time and success rate of PoA.

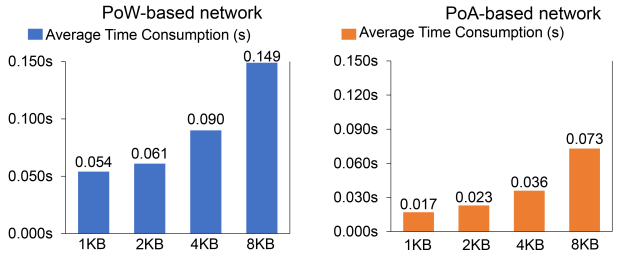


Figure 8. Left: average time consumption for different sizes of messages in seconds for a PoW-based network. Right: average time consumption for different sizes of messages in seconds for a PoA-based network.

D. Performance Evaluation

Consensus Algorithms. As Fig. 7 shows, we set 300, 500, and 700 analog processes to send Data_upload requests to the Ethereum network based on PoW and PoA consensus. We quantify the total response time and success rate of transactions under three different concurrencies. The networks share the same block difficulty, gaslimit, and message size, which are 0x4cccc8, 0xffffffff, and 1 KB respectively.

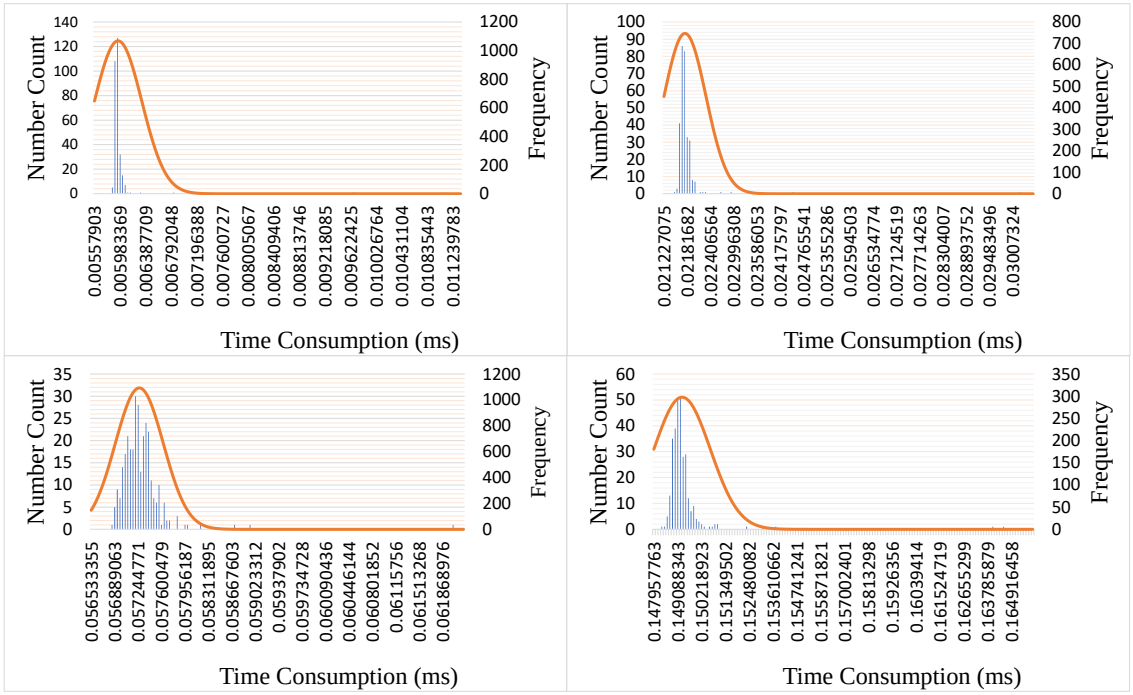
AuthROS maintains excellent interaction whichever consensus algorithm is used. In Fig. 7, the success rate of transactions, which exceeds 99% for both consensus algorithms, is comparable. However, in terms of response time, PoA consensus has obvious advantages over PoW consensus. This difference follows that PoA verifies transactions through preset nodes' voting. At the same time, PoW relies on the mining process to verify transactions. This process needs to consume many computing resources. We then conclude that PoA is more suitable for data transmission in robots.

Messages Size. When robots interact with the contract, the size passed to the contract method would have an impact on the response time. Therefore, we conduct related experiments to study the effect of message size on blockchain networks based on two different consensus algorithms. Message size is set to 4 values of 1KB, 2KB, 4KB, and 8KB. We call the Data_upload interface 300 times through the simulated process and record the average time in Fig. 8.

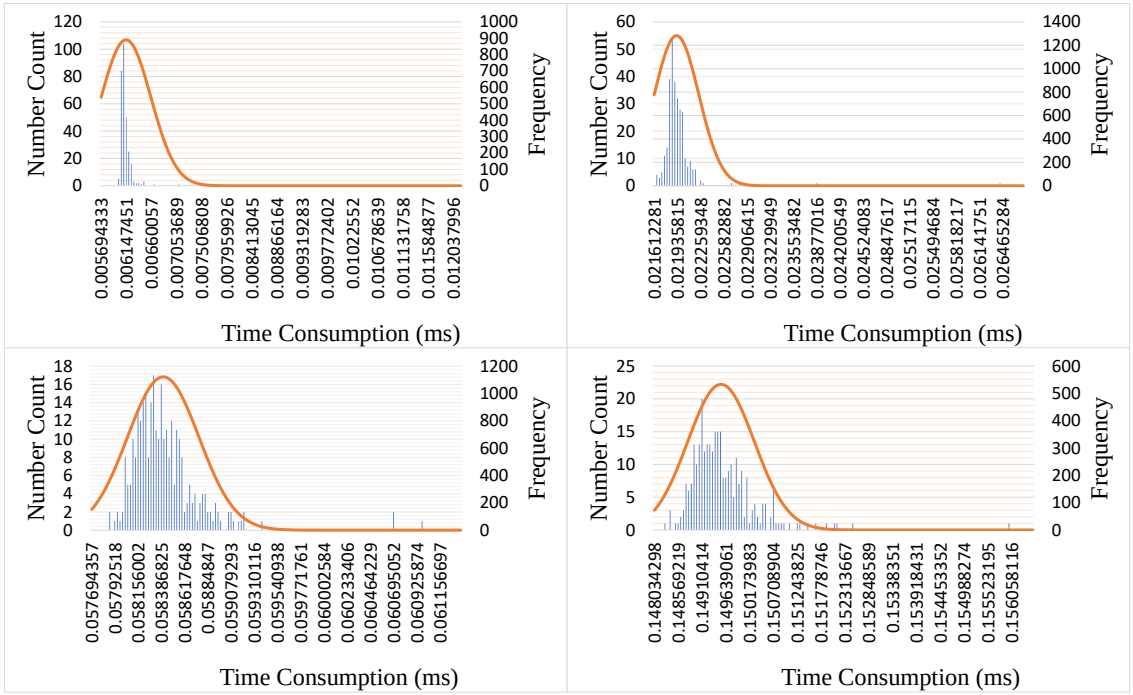
As Fig. 8 depicted, whether the Ethereum network is based on PoW or PoA consensus, the response time grows as message size increases. However, the PoA-based Ethereum network has a shorter overall average response time than the PoW-based Ethereum network. Furthermore, as message size rises, the average response time of the PoA-based Ethereum network grows more slowly.

Efficiency and Stability of SM Algorithm Family. In terms of data communication, AuthROS is equipped with key exchange based on SM2 to ensure the security of the SM4 key. Meanwhile, SM3 is used to generate the hash value of data in a big size. Thus, the efficiency and stability of SM algorithms have a huge impact on the availability and speed of AuthROS.

We conduct experiments on the encryption and decryption speed and stability of SM4 and SM3. We use plain-text data with sizes of 1KB, 2KB, 4KB, and 8KB as encrypted source data for SM4 encrypting and decrypting the data 300 times in Fig. 9 respectively, and recording the average time consumption. We also evaluate the speed and stability of SM3 using an 800 KB matrix as encrypted source data for digest generation 300 times in Fig. 10.



(a) data encryption



(b) data decryption

Figure 9. Time consumption of 300 times data operations in different figure sizes (1KB, 2KB, 4KB, 8KB). The blue bar is for number count, and the orange line is for frequency.

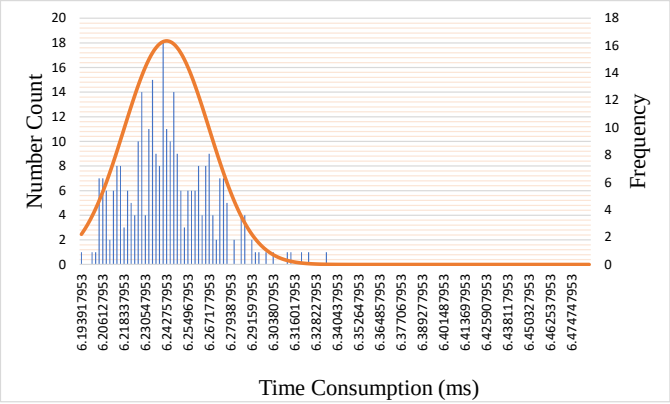


Figure 10. Time consumption of data digest generation each time (800 KB). The blue bar is for number count, and the orange line is for frequency.

The encryption and decryption speeds of SM4 are close, and it is clear that the time consumption of decryption and encryption grows with the data sizes increase, which is due to the features of the symmetric encryption algorithm. However, it can be found in Fig. 9 that no matter the size of data, both decryption and encryption of SM4 have good stability where encryption time consumption concentrates in a certain range. It is the same with SM3. In Fig 10, it is clear that the stability of the SM3 algorithm varies between 6.19ms and 6.34ms.

VI. Conclusion

This paper proposes AuthROS, a novel data sharing framework for ROS, leveraging the Ethereum blockchain and SM algorithms. AuthROS is equipped with a key exchange mechanism and an authority granting mechanism. The key exchange mechanism guarantees the security of the SM4 key used for data encryption, and the authority granting mechanism ensures the trustworthiness of shared data and the controllability of information data. Through systematic experimental evaluation, the security and efficiency of AuthROS are verified. This work is also potential in some other fields as federated learning [34], [35], [38], cloud-edge cooperate robotics [36], [37],smart city, etc.

VII. Acknowledgments

A preprint has previously been published [33]. We thank Songjing Tao (collation of data), Shuang Wu (collation of data), Ming Tang (writing assistance), Zeping Tang (writing assistance), and Zhixuan Liang (acquisition of funding and general support) for their early contributions to this work. Bernie Liu in the preprint is the same author as Boyi Liu in this paper.

References

[1] Quigley M, Conley K, Gerkey B, et al. *ROS: an open-source Robot Operating System*, ICRA workshop on open source software. 2009, 3(3.2): 5.

[2] S Zhang, M Tang, X Li, et al. *ROS-Ethereum: A Convenient Tool to Bridge ROS and Blockchain (Ethereum)*, Security and Communication Networks, 2022.

[3] McClean J, Stull C, Farrar C, et al. *A preliminary cyber-physical security assessment of the robot operating system (ros),* in Proc. of SPIE, 2013, 8741: 874110.

[4] Cheminod M, Durante L, Valenzano A. *Review of security issues in industrial networks*, IEEE transactions on industrial informatics, 2012, 9(1): 277-293.

[5] Dzung D, Naedele M, Von Hoff T P, et al. *Security for industrial communication systems,* in Proc. of IEEE, 2005, 93(6): 1152-1177.

[6] Caiazza G. *Application-level security for robotic networks*, Università Ca' Foscari Venezia, 2021.

[7] White R, Christensen D, Henrik I, et al. *SROS: Securing ROS over the wire, in the graph, and through the kernel*, arXiv preprint arXiv:1611.07060, 2016.

[8] Dieber B, Breiling B, Taurer S, et al. *Security for the robot operating system*, Robotics and Autonomous Systems, 2017, 98: 192-203.

[9] Caiazza G. *Application-level security for robotic networks*, Università Ca'Foscari Venezia. 2021.

[10] Wood G. *Ethereum: A secure decentralised generalised transaction ledger*, Ethereum project yellow paper, 2014, 151(2014): 1-32.

[11] Jiang Y, Shang T, Liu J. *SM algorithms-based encryption scheme for large genomic data files*, Digital Communications and Networks, 2021, 7(4): 543-550.

[12] Lamport L, Shostak R, Pease M. *The Byzantine generals problem*, Concurrency: the Works of Leslie Lamport. 2019: 203-226.

[13] Sundaresan A, Gerard L, Kim M. *Secure ROS*, https://roscon.ros.org /2017/presentations/ROSCon%202017%20SecureROS.pdf, 2017.

[14] Breiling B, Dieber B, Schartner P. *Secure communication for the robot operating system*, in Proc. of SysCon, 2017: 1-6.

[15] Gupta R, Kumari A, Tanwar S, et al. *Blockchain-envisioned softwarized multi-swarming UAVs to tackle COVID-I9 situations*, IEEE Network, 2020, 35(2): 160-167.

[16] Singh P K, Singh R, Nandi S K, et al. *An efficient blockchain-based approach for cooperative decision making in swarm robotics*, Internet Technology Letters, 2020, 3(1): e140.

[17] Ferrer E C, Rudovic O, Hardjono T, et al. *Robochain: A secure data-sharing framework for human-robot interaction*, arXiv preprint arXiv:1802.04480, 2018.

[18] Abichandani P, Lobo D, Kabrawala S, et al. *Secure communication for multiquadrotor networks using Ethereum blockchain*, IEEE Internet of Things Journal, 2020, 8(3): 1783-1796.

[19] Alsamhi S H, Lee B. *Blockchain-empowered multi-robot collaboration to fight COVID-19 and future pandemics*, IEEE Access, 2020, 9: 44173-44197.

[20] Nishida Y, Kaneko K, Sharma S, et al. *Suppressing chain size of blockchain-based information sharing for swarm robotic systems*, in Proc. of CANDARW, 2018: 524-528.

[21] Queralta J P, Westerlund T. *Blockchain-powered collaboration in heterogeneous swarms of robots*, arXiv preprint arXiv:1912.01711, 2019.

[22] Lopes V, Alexandre L A, Pereira N. *Controlling robots using artificial intelligence and a consortium blockchain*, arXiv preprint arXiv:1903.00660, 2019.

[23] Ferrer E C, Jiménez E, Lopez-Presa J L, et al. *Following Leaders in Byzantine Multirobot Systems by Using Blockchain Technology*, IEEE Transactions on Robotics, 2021.

[24] Pacheco A, Strobel V, Dorigo M. *A Blockchain-Controlled Physical Robot Swarm Communicating via an Ad-Hoc Network*, in Proc. of ANTS, 2020: 3-15.

[25] Strobel V, Castelló Ferrer E, Dorigo M. *Blockchain technology secures robot swarms: a comparison of consensus protocols and their resilience to byzantine robots*, Frontiers in Robotics and AI, 2020, 7: 54.

[26] Adam Schmideg, Felix Lange, Aleksandr Sobolev, et al. *Go Ethereum Documentation*, https://geth.ethereum.org/docs, 2022.

[27] Nakamoto S. *Bitcoin: A peer-to-peer electronic cash system*, Decentralized Business Review, 2008: 21260.

[28] Network P O A. *Proof of Authority: consensus model with Identity at Stake*, POA Network, 2017.

[29] Li X, Jiang P, Chen T, et al. *A survey on the security of blockchain systems*, Future Generation Computer Systems, 2020, 107: 841-853.

[30] Adikari J, Dimitrov V S, Mishra P K. *Fast multiple point multiplication on elliptic curves over prime and binary fields using the double-base number system*, Cryptology ePrint Archive, 2008.

[31] J Chen, Y Zhu, D Ye, L Hu, D Pei, G Peng et al. *SM2 elliptic curve public key cryptography algorithm*, http://www.gmbz.org.cn/upload/2018-07-24/1532401673138056311.pdf, 2018.

[32] S Lv, D Li et al. *SM4 block cipher algorithm*, http://www.gmbz.org.cn/upload/2018-04-04/1522788048733065051.pdf, 2018.

[33] Zhang S, Tao S, Li X, et al. *AuthROS: A Secure and Convenient Framework for Data Sharing among Robot Operating Systems (ROS)*, Research Square, 2022.

[34] Liu, Boyi and Wang, Lujia and Liu, Ming. *Lifelong federated reinforcement learning: a learning architecture for navigation in cloud robotic systems*, IEEE Robotics and Automation Letters, 2019, 4(4): 4555-4562.

[35] Liu, Boyi and Wang, Lujia and Liu, Ming and Xu, Cheng-Zhong. *Federated Imitation Learning: A Novel Framework for Cloud Robotic Systems with Heterogeneous Sensor Data*, IEEE Robotics and Automation Letters, 2019, 5(2): 3509-3516.

[36] Liu, Boyi and Wang, Lujia and Chen, Xinquan and Huang, Lexiong and Han, Dong and Xu, Cheng-Zhong. *Peer-assisted robotic learning: a data-driven collaborative learning approach for cloud robotic systems*, 2021 IEEE International Conference on Robotics and Automation (ICRA), 2021, 4062-4070.

[37] Liu, Boyi and Wang, Lujia and Liu, Ming. *ElasticROS: An Elastically Collaborative Robot Operation System for Fog and Cloud Robotics*, https://www.researchgate.net/publication/363032937_ElasticROS_An_Elastically_Collaborative_Robot_Operation_System_for_Fog_and_Cloud_Robotics, 2022.

[38] Zheng, Zhaohua and Zhou, Yize and Sun, Yilong and Wang, Zhang and Liu, Boyi and Li, Keqiu. *Applications of federated learning in smart cities: recent advances, taxonomy, and open challenges*, Connection Science, 2021, 1-28.