Highlights

A Secure Data Sharing Framework for Robot Operating Systems Leveraging Ethereum
Shenhui Zhang, Wenkai Li, Xiaoqi Li, Boyi Liu, Yuqing Zhang, Chunjie Cao

- 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.

- 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).

- 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.
A Secure Data Sharing Framework for Robot Operating Systems Leveraging Ethereum

Shenhui Zhang\textsuperscript{a}, Wenkai Li\textsuperscript{a}, Xiaoqi Li\textsuperscript{a}\textsuperscript{*}, Boyi Liu\textsuperscript{b}\textsuperscript{*}, Yuqing Zhang\textsuperscript{a,c} and Chunjie Cao\textsuperscript{a}

\textsuperscript{a}School of Computer Science and Technology & School of Cyberspace Security, Hainan University, Haikou, China
\textsuperscript{b}Robotics Institute, The Hong Kong University of Science and Technology, Hong Kong, China
\textsuperscript{c}National Computer Network Intrusion Protection Center, University of Chinese Academy of Sciences, Beijing, China

\begin{abstract}
Robot Operating System (ROS) has brought the excellent potential for automation in various fields involving production tasks, productivity enhancement, and the simplification of human operations. However, ROS highly relies on communication but lacks secure data sharing mechanisms. Securing confidential data exchange between multi-robot presents significant challenges in multi-robot interactions. In this paper, we introduce AuthROS, a secure and convenient authorization framework for ROS nodes with absolute security and high availability based on a private Ethereum network and SM algorithms. To our best knowledge, AuthROS is the first secure data-sharing framework for robots loaded with ROS. This framework can meet the requirements for immutability and security of confidential data exchanged between ROS nodes. In addition, an authority-granting and identity-verification mechanism are proposed to execute atomically to ensure trustworthy data exchange without third-party. Both an SM2 key exchange and an SM4 plaintext encryption mechanism are proposed for data transmission security. A data digest uploading scheme is also implemented to improve the efficiency of data querying and uploading on the Ethereum network. Experimental results demonstrate that it can generate a digest from 800KB encrypted data in 6.34ms. Through security analysis, AuthROS achieves secure data exchange, data operations detection, and Node Forging attack protection.
\end{abstract}

\section{Introduction}

The Robot Operating System (ROS) (Quigley et al., 2009; Zhang et al., 2022) 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 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 (Quigley et al., 2015), 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 (McCleart al., 2013; Chemiod et al., 2012; Dzung et al., 2005). First of all, ROS lacks data security protections. By abusing the Publisher-Subscriber mode, an attacker might potentially take data from ROS, putting the data’s immobility and security at risk. In addition, ROS has data integrity issues, and data transferred between ROS nodes based on this architecture can be intercepted or fabricated (Caiazza, 2021). For example, these defects in autonomous driving would result in a serious accident. These two flaws lead to insecure and unstable data exchange and sharing in the scenario of multi-robot interaction based on ROS.

Blockchain (Nakamoto, 2008) is a shared, decentralized data ledger that facilitates the recording of transactions and tracking of assets in business networks. The blockchain utilizes elliptic curve (Knapp, 1992), hashing, and asymmetric encryption algorithms (Alenezi et al., 2020). The elliptic curve algorithms generate users’ addresses, and asymmetric encryption algorithms verify transactions. It also equips data structures and consensus mechanisms. The Merkle tree is a form of data structure (Nakamoto, 2008) in blockchain to maintain data integrity. And consensus mechanisms, such as PoW (Nakamoto, 2008) and PoA (POA, 2017), are significant technology to ensure data consistency on blockchain. Therefore, blockchain has the features of decentralization, openness, autonomy, anonymity, and immutability. In addition, Ethereum (Wood et al., 2014) is a type of blockchain. It offers smart contracts, which are programs running on the blockchain. The Ethereum Virtual Machine (EVM) provided by Ethereum executes the bytecode of smart contracts. The Ethereum account associated with a smart contract is known as a smart contract account (Li et al., 2020).
smart contract’s state allows it to receive and transmit Ether and data. Thus, it can implement more complicated transactions.

As shown in Fig. 1, many attacks exist against the current data sharing in ROS, such as Nodes Forging (Caiazza, 2021), eavesdropping (Sami et al., 2020) and repetition (Puang et al., 2019). Therefore, we utilize the blockchain to solve these drawbacks in ROS framework (Wood et al., 2014) to solve the above drawbacks.

Therefore, to ensure the security of the entire blockchain network and data, we leverage a series of remarkable cryptographic algorithms, i.e., the SM algorithm family (Jiang et al., 2021). Currently, three algorithms are mainly applied: SM2, SM3, and SM4 (Jiang et al., 2021). Based on the SM algorithm family and blockchain technology, we propose AuthROS (Authority in Robot Operating Systems), an efficient and secure data sharing framework for ROS based on Ethereum blockchain technology, Web3, and SM algorithm families. AuthROS allows critical data captured by robots to be shared by other robots with authority within the same Ethereum network.

In addition, the confidential data is maintained for final verification on the Ethereum network. Moreover, it has virtually no restrictions on the types of data. Most types of messages are supported to interact with the Ethereum network. The encryption system based on the SM algorithms may efficiently safeguard the transmission of sensitive data. The advantages of AuthROS are listed below:

1. AuthROS provides uniform and standardized interfaces for ROS- and blockchain-based robots to facilitate communication and collaboration.
2. AuthROS provides adequate security. It can prevent some threats to data in communication, such as Node Forging (Caiazza, 2021).
3. AuthROS has an excellent performance. It can perform data sharing activities swiftly and reliably and it is equipped with SM3 algorithm to facilitate sharing large amounts of data.

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 technical background and Section 3 summarizes the related work. In Section 4, we introduce the framework of AuthROS. Then we formalize our methodology in Section 5 and analyze the security of AuthROS in Section 6. Finally, Section 7 presents the results of the experiments and Section 8 summarizes this paper.

2. Background

In this section, we introduce the technical background of AuthROS, including the ROS system and Ethereum network.

2.1. 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 (Caiazza, 2021). 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 contains a Peer-to-Peer (P2P) network topology. It incorporates asynchronous data flow communication via topics with the synchronous service-based Remote Procedure Call (RPC).

2.1.1. Terms

Each of the software modules is a node (Quigley et al., 2009). Moreover, the nodes communicate with each other by passing messages strongly typed and supporting 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.

2.2. Ethereum

Ethereum (Wood et al., 2014), 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 and hashing algorithms before authenticating transactions.

2.2.1. Truffle Development Environment

An Ethereum resource management channel, development, and testing environment called Truffle are designed to make programming on the Ethereum platform easy. Truffle can construct, link, and distribute smart contracts. Additionally, contract compilation binary files may be managed using Truffle. Automated scripts incorporated into Truffle ensure that nodes can only communicate with one another over channels designed for them.
Table 1
Comparison of Our Study and Other Related Literature. ‘Integrates’ is the convergence of consensus algorithms and different technologies or hardware devices. ‘SM’ represents ‘Supports Multirobots’. SRC means ‘Swarm Robot Communication’, SDS means ‘Secure Data Sharing’, and RS means ‘Robot Security’.

| Reference                      | Integrates | Others                | SM | Classification |
|--------------------------------|------------|-----------------------|----|----------------|
|                                | Consensus  | Others                |    | SRC | SDS | RS |
| ferrer et al. (2018)           | PoW        | Federal Learning      | ✓  | ✓   | ✓   | ✓  |
| gupta et al. (2020)            | PoW        | 6G Network            | ✓  | ✓   | ✓   | ✓  |
| singh et al. (2020)            | PoA        | Game                  | ✓  | ✓   | ✓   | ✓  |
| abichandani et al. (2020)      | PoA & PoA  | IoT                   | ✓  | ✓   | ✓   | ✓  |
| sen et al. (2022)              | PoW        | Task Assignment       | ✓  | ✓   | ✓   | ✓  |
| nishida et al. (2018)          | PoW        | IPFS                  | ✓  | ✓   | ✓   | ✓  |
| queralta and westerdeld (2019) | PoW & PoS  | Heterogeneous Robots  | ✓  | ✓   | ✓   | ✓  |
| lopes et al. (2019)            | PoW & PoS  | Oracle                | ✓  | ✓   | ✓   | ✓  |
| pacheco et al. (2020)          | PoW        | Pi-pucks Robots       | ✓  | ✓   | ✓   | ✓  |
| strobel et al. (2020)          | PoW & LCP & W-MSR | ARGoS Simulator | ✓  | ✓   | ✓   | ✓  |
| ferrer et al. (2021)           | PoA        | Byzantine             | ✓  | ✓   | ✓   | ✓  |
| roy et al. (2022)              | PoW        | Cloud Framework       | ✓  | ✓   | ✓   | ✓  |
| our previous study             | PoW & PoA  | SM Algorithms & UDP   | ✓  | ✓   | ✓   | ✓  |
| our present study              | PoW & PoA  | SM Algorithms & TCP & Security Protocols | ✓ | ✓ | ✓ | ✓ |

2.2.2. **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 (Adam et al., 2022). In this paper, the Geth is used to build an Ethereum private network. Ethereum supports Externally Owned Accounts (EOA) and smart contract accounts. Except for 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.

2.2.3. **Consensus Algorithm**

The consensus algorithm provides the immutability, automation, and anonymity of blockchain transactions. It maintains the meaning and value of blockchain technology as a distributed database, which ensures that the states of the blocks on the chain remain consistent. Proof of Work (PoW) (Nakamoto, 2008), Proof of Authority (PoA) (POA, 2017) 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.

2.2.4. **Smart Contract**

The smart contract is an executable software program that can be interacted with peers on the network (Li et al., 2020). 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 and multi-robot systems.

3. **Related Work**

The large-scale application of robots will inevitably involve many problems, two of which are the security of data transmission and sharing and the classical Byzantine problem (Lamport et al., 2019). Due to its superior security performance, blockchain technology has been favored by researchers in the robot community. In Table 1, much research on integrating robots and blockchain has been carried out.

Swarm communication has been the subject of several studies. To combat COVID-19 and break through the bottleneck of existing multi-swarming UAVs based on 5G, Gupta et al. (2020) propose a blockchain-envisioned software multi-swarming UAV communication scheme based on a 6G network with intelligent connectivity. To overcome the constraints of existing robotic control and communication techniques, Singh et al. (2020) offer an efficient communication framework for swarm robotics based on PoA consensus. Ferrier et al. (2018) introduced the first learning framework for secure, decentralized, and computationally efficient data and model sharing among multiple robot units installed at multiple sites. Abichandani et al. (2020) used a set of experiments to validate that Ethereum can be a secure media for communication for multiple small Unmanned Aerial Vehicles (sUAVs). ŞEN et al. (2022) propose a novel mechanism for assigning tasks. It constructs a task assignment network using smart contracts, distributing communication between clusters of several robots. Roy et al. (2022) utilizes blockchain technology to connect numerous cloud servers, cloud-based machine learning inference, and global path planning for IoT devices such as sUAVs.
Secure information sharing is also an essential topic of research. Alsamhi and Lee (2020) propose a framework to facilitate information sharing within multi-robot through the Ethereum blockchain. This framework is proved to be effective. Nishida et al. (2018) introduce a methodology for sharing information amongst autonomous robots and demonstrate through experiments how the differences in data size stored on the blockchain affect the chain length. Queralta and Westerlund (2019) present a novel approach to managing collaboration terms in heterogeneous multi-robot systems with blockchain technology. This method can estimate the computational capabilities of various robots. In addition, different robots can be integrated with environmental data to evaluate and rank the indicators, such as the quality and precision of each robot’s sensor data. Lopes et al. (2019) propose an architecture that uses blockchain as a ledger and smart-contract technology for robotic control by using external parties, Oracles, to process data. The proposed architecture shows excellent potential for secure information sharing between robots.

There comes the classic Byzantine problem with Robotic swarms. In the survey of Ferrer et al. (2021), a set of Byzantine Follow The Leader (BFTL) problems were presented, and based on blockchain technology, algorithms to tackle the BFTL problems were proposed too. Pacheco et al. (2020) presented a robot swarm composed of Pi-puck robots that maintain a blockchain network. The blockchain serves as a security layer to neutralize Byzantine robots. Strobel et al. (2020) demonstrated how robotic swarms achieve consensus in the presence of Byzantine robots utilizing blockchain.

In our previous work (Zhang et al., 2022), we developed an efficient ROS-Ethereum platform, which encrypted and sent data using SM algorithms and UDP protocol. To achieve more secure data sharing that can resist attacks without sacrificing performance. We present AuthROS, which uses TCP/IP rather than UDP, and customizes security protocols and algorithms in Section 5.

4. AuthROS Framework

This section explains the AuthROS framework for data sharing, which is based on blockchain technology, the SM algorithm family, and ROS. The core concept of AuthROS is to leverage smart contracts, consensus mechanisms, and encryption technology to assure the data security of the entire data generation, transit, and sharing process.

4.1. Assumptions and Design Goals

4.1.1. Assumptions

To limit the scope of our research and ensure the availability and efficacy of AuthROS, we incorporate some fair assumptions into its design. These assumptions will heavily influence the role of AuthROS. The following assumptions are made, including (1) Blockchain security, (2) Robots manager, (3) Identity knowability, (4) Unique means of sharing, and (5) Administrator.

(1) **Blockchain security.** The most notable characteristics of blockchain technology are distributed computing and an integrated encryption system. Due to these properties, blockchain possesses superior security performance. We, therefore, assume that Ethereum, as an information-sharing channel, is highly secure and reliable.

(2) **Robots manager.** Every robot or cluster employing AuthROS for data communication will have at least one manager. These administrators will be the Core Users of AuthROS (CURA). In addition, CURA performs data sharing rather than the robot.

(3) **Identity knowability.** After establishing the trust relationship, different CURAs only use AuthROS for data sharing. In other words, the premise of data sharing between CURAs is mutual familiarity and trust. Any CURA will not disclose the EOA address that guarantees its identity to any third party.

(4) **Unique means of sharing.** CURAs will only share information via AuthROS. It is crucial for AuthROS accessibility.

(5) **Administrator.** Regardless of the established consensus mechanism, the blockchain should have a manager to deal with the growth of its members and other emergencies.

4.1.2. Design Goals

AuthROS is designed to accomplish the following objectives, including (1) Efficiency, (2) Traceability, (3) Data preservation, (4) Anti-attack, (5) Large scope, (6) High availability and (7) Controllability.

(1) **Efficiency.** Data encryption, parsing, uploading, and sharing should consume as little time as possible.

(2) **Traceability.** When data sharing is not performed properly or data is not recorded normally, the administrator can acquire signals about out-of-the-ordinary activities to determine if rules have been violated or an attacker is present.

(3) **Data preservation.** Safeguards should be in place to prevent unrelated third parties from stealing and misusing shared data.

(4) **Anti-attack.** AuthROS must be able to resist certain attacks like Nodes Forging (Caiazza, 2021).

(5) **Large scope.** The framework should be able to provide data-sharing services for a variety of robot individuals or groups.

(6) **High availability.** Individuals or groups of robots running different versions of ROS can share information.

(7) **Controllability.** The overall process of data sharing should be controllable, as should the authority of any CURA to access the data shared by other CURA.
4.2. System Model

Users must first upload a "name/token" key-value pair, an SM4 key, and an SM2 public key necessary to authenticate the digital signature. Identity Check and Authority Grant depends on the information. After registering the user’s identification, the user can choose a topic to monitor. The monitored topic often forwards some essential information, such as the data in radar, camera, and other sensors. Once the topic delivers data, AuthROS will immediately start a subscriber to capture and parse the topic’s contents. The data will be delivered to the blockchain network management module of AuthROS after being encrypted by SM4 and signed by SM2. After the SM2 signature has validated the data ciphertext, the SM3 hash method will generate 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. Authority Grant is one of the key mechanisms explained in Section 5. The system’s entire operation is depicted in Fig. 2. This structure possesses the following characteristics, (1) Plasticity and (2) Process security.

(1) Plasticity. The AuthROS blockchain network uses a private chain, which has larger data throughput advantages than the public chain and is more flexible regarding block time and consensus conditions. Moreover, the semi-decentralized structure of the private chain makes adding new members to the network more convenient.

(2) Process security. SM4 is the symmetric encryption technique AuthROS to encrypt all data transmission and interaction operations. When the Data Digest Interaction System is combined with Identity Check, and Authority Grant Mechanism, data integrity, security, and tamperability can be guaranteed. Moreover, AuthROS can resist some attack methods. It will be explained in Section 6.

4.3. Data Sharing Protocol

In this section, we will introduce the core mechanism in AuthROS. Including key distribution protocol, data encryption scheme, etc. Likewise, due to the possibility of intercepting and altering data during transmission, AuthROS proposes stricter standards for data integrity and security. Consequently, AuthROS devised a data digest-based interaction strategy based on the SM2 digital signature algorithm and the SM3 hash algorithm. The server locally stores the whole data, and the blockchain network uploads the data digest. The notations used in the design are summarized in Table 2.

4.4. Preliminaries

The private Ethereum blockchain chain is suited as a platform for information sharing in AuthROS due to its immutability, semi-decentralization, and anonymity. The blockchain network will then be described in depth.

4.4.1. Blockchain Network

We pick 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, (1) Same contract and (2) Same block difficulty and quantity of users.

(1) Same contract. Regardless of the type of consensus mechanism the network employs, we utilize the same contract so that the internal interfaces are consistent.

(2) Same block difficulty and quantity of users. The blockchain network based on the two consensuses mim-
Table 2  
Summary of Notations Used in This Paper.

| Notation | Meaning |
|----------|---------|
| AU_i     | The ith user of AuthROS |
| T        | Data type of ROS topic transmission specified by the user |
| LV_{x,y,z} | Three axis velocity of data of odometry in ROS |
| AV_{x,y,z} | Triaxial angular velocity of odometry in ROS |
| Pose     | Pose information contained in data of odometry in ROS |
| Cov      | Covariance information contained in data of odometry in ROS |
| TS       | Timestamp information contained in odometry in ROS |
| T        | Time of ROS data captured |
| CR       | Cloud server of AuthROS deploying Ethereum blockchain network |
| ND       | Plaintext data |
| CT       | Ciphertext of plaintext |
| Addr_i   | Address of the identity of the ith user in the Ethereum network |
| d        | Structure for users to store shared data in Ethereum network |
| σ        | Structure for users to store their keys in the Ethereum network |
| v        | Structure for users to store keys of other users in Ethereum network |
| EU_i(d,σ,v,Addr_i) | Corresponding identity of the ith AuthROS user in Ethereum network |
| N/d      | Username/password pair |
| k_i      | A set of SM2 public keys published regularly |
| (P_S^e,d_S^e) | The SM2 public/private key pairs published regularly of the system |
| K^e_C | User’s SM4 key |
| (P^e_C,d^e_C) | Public/private key pair for signing and verifying user data |
| (r,s)   | SM2 signature value |
| SM2E :  | The process of generating ciphertext |
| {ND,P_S^e} → CT | CT by encrypting plaintext data ND with SM2 key P_S^e |
| SM2D :  | The process of decrypting CT using |
| {CT,P^e_C} → ND | d^e_C to get ND |
| SM4 :  | The process of generating data |
| {ND,K_C} → Ciphertext CT by encrypting plaintext data ND with SM4 key K_C |
| SM2S :  | The data to be sent is signed with |
| {M^{radw},d^e_C} → (r,s) | SM2 signature algorithm |

Regarding smart terminal devices due to the limited computing resources. We build our private Ethereum network in the cloud server to address this problem. At the same time, the robots can obtain the connection to the blockchain of AuthROS through the local ROS master running on the host with internet access.

Due to the robot’s limited processing power, building an Ethereum node there will encounter the computational bottleneck typical of Edge Computing. Therefore, the Ethereum network composed of 3 nodes was 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. Fig. 3 depicts Ethereum’s private chain architecture in AuthROS.

The blockchain network is deployed in the cloud server and is composed of three nodes: Miner Node is the boot node. The overlay network bootstrap node refers to the node that serves as the initial connection point for the network. The Miner Node is interconnected with nodes such as node1 and node2. Node2 and Node3 host EOA accounts that all local ROS robots use to connect to the Ethereum network. These robots will be bound with an Externally Owned Account (EOA) which is necessary due to the demand for interaction with blockchain in Ethereum.

4.4.2. Genesis Block

The genesis block saves some significant parameters which lay the foundation for the whole blockchain network. To initialize a private Ethereum network, a genesis file that includes these critical parameters written in JSON is needed. Fig. 4 depicts the file we used to initialize a PoW-based Ethereum network.

Some key parameters include (1) ChainID, (2) GasLimit, (3) Difficulty and (4) Alloc.

(1) ChainID. Using this field, a local private blockchain network can connect to other nodes on the blockchain.
A Secure Data Sharing Framework for Robot Operating Systems Leveraging Ethereum

(2) **GasLimit.** It limits the value of gas used in a single transaction. If the gas value is exceeded, the transaction stops. In this network, the gas price, denoted in GWei, is the amount that the transaction sender’s acceptable price. Ether is the native cryptocurrency used to pay for gas (Zhang et al., 2022).

(3) **Difficulty.** This field indicates the miner’s computational difficulty in producing the next block. It is proportional to the number of computational resources used. The higher the difficulty is set, the longer it takes to find blocks (Zhang et al., 2022).

(4) **Alloc.** This field is used to specify the amount of cryptocurrency in wei associated with particular accounts. More importantly, it is just exclusive to the private chain. To enable each node in the network to operate normally, we set Alloc to be greater than Gaslimit (Zhang et al., 2022).

To evaluate the effect of different consensus algorithms on blocking time and verification time of transactions, we also prepare a genesis file of the PoA-based Ethereum network depicted in Fig. 5. And for the effect of difficulty on the verification time of transactions, the block difficulty values are set to the following values in both of the genesis files.

- **Difficulty 1:** 0x1
- **Difficulty 2:** 0x100
- **Difficulty 3:** 0x10000
- **Difficulty 4:** 0x1000000

Two consensus algorithms are evaluated for AuthROS. The two networks based on these algorithms (PoA and PoW) share the same general configuration in the genesis file.

**4.4.3. Hardware Platform**

The robots we used came equipped with a Jetson Nano B01 (Quad-core ARM A57 64-bit @ 1.43Ghz 4GB LPDDR4-3200), a controller with a built-in 9-axis IMU sensor, RPLIDAR A1 radar, a Wi-Fi module that can provide up to 867 mbps 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) were combined to build a complete map of the closed experimental space. ROS Melodic was set up in every robot, as long as the personal computer was running ROS Master. Each robot can connect to the ROS Master through the Wi-Fi module to realize stable communication. The personal computer running ROS Master may also connect to the server hosting the Ethereum network, allowing robots to interact with Ethereum.

The controller and Jetson Nano were connected through a UART connection using a software function call provided by the controller’s onboard C++ SDK. the controller collected the 9-axis IMU sensor and motor data. The RGB-D and RPLIDAR were connected to the Jetnano to capture images and collect lidar data. Motion commands were 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).
4.4.4. Proposed Data Sharing Protocol

The design of AuthROS consists of three phases: initialization, data upload, and query. Fig. 6 depicts the subprocess description corresponding to each level. Next, we will provide an overview of the whole procedure, followed by a discussion of essential mechanisms in Section 5, including (1) Key derivation (generation), (2) Key allocation, (3) Register, (4) Topics set and (5) Encryption & transfer.

1) **Key derivation (generation).** Through the whole data sharing life cycle, its validity, integrity, and immutability must be verified. We must thus utilize public key cryptography to generate a pair of asymmetric keys for each user and sign their shared data. To satisfy the conditions mentioned above, 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 (Jianhua et al., 2012). You need provide your random number private key \( k \) and AuthROS will compute the public key \( P_C = [d_C]G \) using the multiple points fast algorithm of multiple elliptic curves, where \( G \) is the base point of the elliptic curve, and its order is a prime number. In the meantime, AuthROS will also employ the SM2 key derivation function to build the appropriate public/private key pair \((P_s, P_{C}^i)\) for key exchange. Specific algorithms, which will not be explained here, can be found in the standard for SM2 algorithms (Jianhua et al., 2012).

2) **Key allocation.** Under the current TCP/IP network transmission architecture, data is typically transmitted in plaintext over the transmission link, making it easy for malicious users of a third party to intercept the data and launch a series of attacks such as replay attacks and man-in-the-middle attacks, thereby jeopardizing system security. 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 that the SM4 key for encrypting plaintext data and the SM2 public key for confirming the digital signature can be securely delivered to the blockchain for storage.

First, users must enter their own SM4 key \( K_C \) and SM2 public key \( P_C \). Then, user must choose a system public key \( P_S \) to encrypt the message \( SM2E : \{ (K_C^i, P_C^i), P_S^i \} \rightarrow CT \), where \( d_S^i \) and its associated \( P_S^i \) form a public/private key pair \((P_s^i, d_S^i)\), \( k \) is a set of SM2 public keys regularly published by us. In addition, user must use their personal SM2 private key \( d_C^i \) to sign the resulting ciphertext \( CT, SM2S : \{CT, d_C^i\} \rightarrow (r, s) \). Then, add an instruction frame after the ciphertext \( CT \) to produce the final message \( \{CT, (r, s), P_S^i, Command\} \), where \( Command \) represents the instruction of operations that 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 \( P_S^i \), \( SM2D : \{CT, d_S^i\} \rightarrow (K_C^i, P_C^i) \) after receiving the data packet. After decryption, \( CR \) will utilize \( P_C^i \) to validate the ciphertext’s authenticity and integrity using \( CT, SM2V : \{CT, P_C^i, (r, s)\} \rightarrow True / False \). After the integrity and accuracy checks, the keys \( K_C^i \) and \( P_C^i \) will be saved in the blockchain network for future operations.

3) **Register.** At this stage, user \( AU \) must register with AuthROS and upload a key-value pair \( \{N_i, t_i\} \) as their iden-
Figure 7: The Workflow of ROS in Local Robots. Node1, Node2, and AuthROS Node are on the Ethereum network. Master represents the ROS system. Solid lines and arrows indicate the relationships of function calls between nodes.

5. Security Protocol and Algorithms

In this section, we will expand the security protocols in AuthROS in detail, such as (1) Identity Check, (2) Authority Grant, (3) Digest Interaction & Query, (4) Blockchain Control, (5) Longevity of AuthROS Blockchain Network and (6) Security of the AuthROS Blockchain Network. Additional symbols are displayed in Table 3.

5.1. Identity Check

The identity verification module further guarantees the integrity of transmitted data and the validity of users. In the encrypted transmission phase, the basic format of the data packet is \[ \{ ..., CT_1, (r, s), Command, t_1 \} \text{CT}_2, N_i \}, \] where \((r, s)\) is the signature value generated by the data sender \(N_i\) using its own SM2 private key \(d_{C_i}\) to sign the ciphertext \(CT_1\). \text{Command}\ is the identification indicating the operation of the ciphertext \(CT_1\). If \(N_i\) is the user’s name and \(CR\) for all incoming packets \(N_D\), the packet will be parsed to acquire \(N_i^{\text{recv}}\) and the mapping \(\text{Map}(N_i^{\text{recv}})\) will then be queried. If \(\text{Map}(N_i^{\text{recv}}) = \text{Null}\), it indicates that the packet’s sender has not registered his identity in AuthROS, \(CR\) will refuse to proceed to the next phase of packet processing.
format is incorrect, it signals that the data packet is susceptible to tampering and replacement, and if the format is valid, it will refuse to proceed with the next stage of processing the data packet. If the format is valid and the verification succeeds, the ciphertext $CT_i$ will be processed according to the command $Command$. After proving the data sender's identity, $CR$ decrypts the ciphertext $CT_i$ to retrieve information such as $T$, $Type$. SM3 cryptographic hash technique is used to hash ciphertext $CT_i$ to yield data digest $SM3 : \{CT_i\} \rightarrow D_i$. Supposing there are $n$ data types shared in the network, we number each type, producing $[SRtype]$. Then, required information frames are packaged as $\{D_1, SRtype, T\}_d$ and decrypted by the identity address of user $AU_i$ on the chain $EU_i, Addr_r$. Simultaneously, the entire ciphertext data will be cached in $Cache$ to create mapping $Map \{SRtype, T\} \rightarrow CT_i$.

5.2. Authority Grant

Users must be allowed by the data owner before querying the data supplied by other users, as this is how AuthROS's trustworthy data sharing is primarily implemented. This approach efficiently ensures the security of information sharing and can guarantee that no third parties are involved in the secure data exchange. Each user $AU_j$ who has successfully registered with AuthROS has an identity $EU_j$ in the AuthROS blockchain network. If $AU_j$ wants another registered user $AU_j$ to access his shared data, he (or she) can authorize $AU_j$ via the AuthROS identity authorization module. User $AU_j$ must first collect the account address $Addr_r$ of user $AU_j$ in the blockchain network before sending packet $\{\{Addr_r, (r, s), Command, t_i\} \rightarrow CR \}$. After verifying the packet's identification, the smart contract interface Authority Grant is automatically executed, attaching the SM4 key $K_v$ stored in the identity $EU_j, \sigma$ on the chain of user $AU_j$ to the identity $EU_j, \sigma$ on the chain of $AU_j, EU_j, v = EU_j, \sigma$. Thus, user $AU_j$ can easily access the ciphertext data shared by user $AU_i$ via the key of user $AU_i$ stored in $EU_j, v$, and decrypt the ciphertext using the key.

5.3. Digest Interaction & Query

The AuthROS data digest interaction is primarily divided into two aspects: data upload and queries. Now we shall elaborate on these two facets, (1) Data upload and (2) Data query.

(1) Data upload. We take the Odometry-type data packet $\{\{CT_1, (r, s), Command, t_i\} \rightarrow CR \}$ as an example. After proving the data sender’s identity, $CR$ decrypts the ciphertext $CT_i$ to retrieve information such as $T$, $Type$. SM3 cryptographic hash technique is used to hash ciphertext $CT_i$ to yield data digest $SM3 : \{CT_i\} \rightarrow D_i$. Supposing there are $n$ data types shared in the network, we number each type, producing $[SRtype]$, and map the data type $Type$ of $CT_i$ to the $SRtype : Type \rightarrow SRtype$. Then, required information frames are packaged as $\{D_1, SRtype, T\}_d$, adding $D_1$. The data digest is then uploaded to the blockchain in order $\{D_1, SRtype, T\} \rightarrow EU_i, d$ according to the identity address of user $AU_i$ on the chain $EU_i, Addr_r$. Simultaneously, the entire ciphertext data will be cached in $Cache$ to create mapping $Map \{SRtype, T\} \rightarrow CT_i$.

(2) Data query. When user $AU_j$ request his own shared data, he must first use $EU_j$ and then, after querying the blockchain network’s data $\{D_1, SRtype, T\}_d$, use $\{SRtype, T\}$ to query the ciphertext data contained in $Cache$, $\{SRtype, T\} \rightarrow CT_i^{cache}$. Then, we will calculate the hash value of $CT_i^{cache}$ using the SM3 hash method $SM3 : \{CT_i^{cache}, k_3\} \rightarrow D_i^{cache}$. Compare $D_i^{cache}$ with $D_i$ next. If $D_i^{cache} \neq D_i$, it indicates that the data was corrupted during transmission and alerts the user $AU_i$ that the query was unsuccessful. If $D_i^{cache} = D_i$, user $AU_j$ will decrypt ciphertext $CT_i^{cache}$ with the appropriate key to acquire the plaintext data. It is worth saying that if user $AU_j$ needs to
query data shared by other users, say, \( AU_i \), it is required to guarantee that user \( AU_i \) has authorized user \( AU_j \) that \( EU_{i,v} \) contains \( EU_{i} \).

This method is described by Algorithm 1. The advantages of this data digest interaction technique are as follows, (1) Superior security and (2) Efficiency.

1. **Superior security.** Digital signature, in conjunction with the data digest interaction scheme and the exclusive tamper-proof of blockchain technology, may effectively assure the integrity and security of the data transmission and storage processes.

2. **Efficiency.** The contract interface parameter size significantly impacts the data uplink speed. Using a data digest with a set length and merely 16 bytes as a parameter can significantly increase the data uplink speed.

**Algorithm 1: Digest Interaction in AuthROS**

**Data:** Ciphertext \( CT_1 \), Timestamp when the data was captured \( T \), data type \( S\text{Type} \), identity of data sender on the chain \( EU_i \), the SM4 key \( K_c^i \) of \( EU_i \)

**Result:** Whether or whether the data has been correctly uploaded or returned to the correct user

\[
\begin{align*}
\text{begin} & \quad \text{(a)}\text{ Process of data uploading} \\
& \quad D_1 = SM3: \{ CT_1 \} \\
& \quad S\text{Type} : Type \rightarrow S\text{Type} \\
& \quad EU_i.d = \{ D_1, S\text{Type}, T \} \\
& \quad \text{Cache} : \{ S\text{Type}, T \} \rightarrow CT_1 \\
\text{end} \\
\text{begin} & \quad \text{(b) Process of data query} \\
& \quad \text{Data} = EU_i.d \\
& \quad CT = \text{Cache} \{ S\text{Type}, T \} \\
& \quad D_2 = SM3: \{ CT \} \\
\text{end}
\]

\[ \text{if } D_2 = \text{Data}.D_1 \text{ then} \]

\[ \quad \text{return the } CT \text{ to the inquirer} \]

**5.4. Blockchain Control**

We have established a set of guidelines for the distribution of Gas (Caiazza, 2021) and the maintenance of network accounts. These regulations contribute to the security and longevity of the AuthROS blockchain network. In Section 6, we will explain how its effect can be performed.

**5.5. Longevity of AuthROS Blockchain Network**

In the Ethereum network, the presence of transactions is one of the most significant indicators of network health. Moreover, the data exchange will inevitably result in Gas consumption. However, AuthROS transactions can indicate if data is exchanged. To enhance the user experience on the AuthROS network, we have constructed a Gas distribution scheme to reduce Gas consumption. In this scheme, we stipulate that users will receive Gas whenever other users query their network-shared data. The number of Gas incentives will vary based on the rate at which network-wide data is transmitted. To determine the number of rewards and costs, we use the Algorithm 2 for evaluating value. We count \( \{ Num_{SR\text{Type}} \}_{T1} \) at a \( T1 \) interval. Each time a specific type of data is queried, the function \( \text{Count}(SR\text{Type}) \) is called.

\[
W_{SR\text{Type}} = \frac{\{ Num_{SR\text{Type}} \}_{T1}}{Num_{T1}}
\]

Then, each \( W_{SR\text{Type}} \) is added to array as the Equation (2).

\[
\text{Matrix } \text{Num} = \begin{pmatrix} W_1 & W_2 & \cdots & W_n \end{pmatrix}
\]

As the Equation (3), Matrix sharing will additionally record the number of times the data shared by the \( i_{th} \) user in the network has been requested.

\[
\text{Matrix } \text{Sharing} = \begin{pmatrix} \{ Num_1 \text{Sharing} \}_{T1} \\
\{ Num_2 \text{Sharing} \}_{T1} \\
\vdots \\
\{ Num_n \text{Sharing} \}_{T1} \\
\end{pmatrix}
\]

As each data query operation would rise dynamically owing to chain length and other factors, we use \( GAS \) as the unit of gas reward. Equation (4) shows the final gas reward for the user.

\[
\text{Matrix } \text{Num} \times \text{Matrix } \text{Sharing} \times GAS
\]

\[
\text{5.6. Security of AuthROS Blockchain Network.}
\]

Even though the PoA consensus algorithm has significantly improved the scalability and efficiency of PoA-based blockchain networks, there are still security flaws. For instance, when the verifier attacks the network, it can cause irreparable damage.

To address the issues above, we propose a block verifier election mechanism based on contributions. Block verifiers are elected at the beginning of each round of verifier election by the Algorithm 3. It depends on the value of the data users send and receive from the network.

The value of the data that users send to the network is comparable to the algorithm for evaluating value. The calculation in Equation (5) yields a value.

\[
\text{Value}^i_{\text{Output}} = \text{Matrix } \text{Num} \times \text{Matrix } \text{Sharing}^i
\]

The value of the data collected by the user from the network is calculated as the opposite of the output value. Input each
Algorithm 2: Value Assessment in AuthROS

**Data:** The serial number of different data types [SRtype], request cache WR where keep all query requests happened in TI, the length of user list of AuthROS

**Result:** Gas reward rewardi obtained by the ith user in interval TI

```plaintext
while Len(WR) do
  Read a data packet from WR and retrieve the corresponding SRtype and the target of query operation EUi.
  if SRtype ≠ null then
    NumSRtype = 1
    NumSRtypei = 1
    Len(WR) = 1
  end
  for i in Len(SRtype) do
    Wti = Num 
    MatrixNumi = (W1, W2, ..., Wn)
  end
  for i in L do
    MatrixSharing = Num
    rewardi = MatrixNum × MatrixSharing × GAS
  end
end
```

Calculate the overall worth of all data in the network with the Equation (9).

\[
W = (W_1, W_2, ..., W_n) \times \text{Matrix Num}
\]

as well as the contribution ratio of each user \( C_{\text{Input}}^i \), \( C_{\text{Output}}^i \) in the Equation (10).

\[
C_{\text{Input}}^i = \frac{\text{Value}_{\text{Input}}^i}{W}, C_{\text{Output}}^i = \frac{\text{Value}_{\text{Output}}^i}{W}
\]

Finally, the block verifiers are elected based on the verifier election conditions with the Equation (11).

\[
\text{Verify} = \left\{ \begin{array}{l}
  \text{abs}(C_{\text{Output}}^i - C_{\text{Input}}^i) \\
  \text{abs}(C_{\text{Output}}^i - C_{\text{Input}}^i) \\
  \text{Max}(C_{\text{Output}}^i \times C_{\text{Input}}^i)
\end{array} \right.
\]

This election condition specifies the selection of all qualifying user AUi whose absolute value \( C_{\text{Output}}^i - C_{\text{Input}}^i \) is within \( 0-5\% \), \( 5\%-10\% \), \( 10\%-15\% \) respectively. If no qualifying user exists in the interval, no verifier will be chosen in the interval. The user who has the smallest value \( C_{\text{Output}}^i - C_{\text{Input}}^i \) within these intervals are chosen to create the verifier sequence [Verify] for this round TI. Algorithm 3 describes this process in depth. In the Section 6, the security performance of this method will be detailed.

6. Security Analysis

In this section, we analyze our scheme, AuthROS, with the following security features: (1) Blockchain security, (2) Attack resistance and (3) Data preservation.

**Theorem 6.1. Blockchain security.** The blockchain network verifier for AuthROS will not cause the network to be inaccessible.

**Proof.** AuthROS provides contribution-based block verifier election. It can ensure that the selected block verifier will not launch an attack or cause network paralysis. The purpose of the first condition of the algorithm’s election conditions is to pick the network members who provide and request data value. Based on the first criterion, the second condition selects the member who contributes a greater product of data value and data value. The primary purpose of establishing these two requirements is to limit the value contribution and claim of elected block verifiers to a predetermined range. Such that they are neither excessively large nor small. For \( C_{\text{Input}}^i \) and \( C_{\text{Output}}^i \), \( \sum_{i=1}^{n} C_{\text{Input}}^i + \sum_{i=1}^{n} C_{\text{Output}}^i = 1 \). For example, if \( C_{\text{Output}}^i \) is too large, \( C_{\text{Output}}^i > 50\% \), then the value given by user \( AU_i \) to the network exceeds 50 percent of the overall value of the entire internal data transaction. It means that the majority of other network users rely on the data shared by user \( AU_i \). Once the user \( AU_i \) becomes...
Algorithm 3: Verifier Election in AuthROS

Data: the serial number of different data types [SRtype], web request cache WR where keep all query requests happened in TI, the length of user list of AuthROS L

Result: Verifier list [Verify]

while Len(WR) do
    Read a data packet from WR and retrieve the corresponding SRtype, the target of query operation EUjShare and the request sender EUjInquirer
    if SRtype ≠ null then
        NumjSRtype + = 1
        NumjSRtype + = 1
        Len(WR) = = 1
    end
    for i in Len(SRtype) do
        MatrixjNum=(W1,W2...,Wn)
    end
    for i in L do
        Matrix Sharing = \[
        \begin{pmatrix}
        NumjSRtype & EUjShare \\
        Num2 & EUjShare \\
        \vdots & \vdots \\
        Numn & EUjShare \\
        \end{pmatrix}
        \\
        Matrix Input = \[
        \begin{pmatrix}
        NumjSRtype & EUjInquirer \\
        Num2 & EUjInquirer \\
        \vdots & \vdots \\
        Numn & EUjInquirer \\
        \end{pmatrix}
        \\
        CijInput = \frac{MatrixjNum × Matrix Sharing}{W}
        \\
        CijOutput = \frac{MatrixjNum × Matrix Input}{W}
    end
    for i in 3 do
        [ Verify ] = \[
        \begin{cases}
        \text{abs} \left( CijOutput - CijInput \right) & \text{if } \left( CijOutput \times CijInput \right) \\
        \text{Max} \left( CijOutput \times CijInput \right) & \text{else}
        \end{cases}
        \]
    end
end
example, user $AU_j$ must grant user $AU_i$ access to his or her shared data. The following steps must be completed.

1. When user $AU_j$ is disconnected from the network environment, he (or she) acquires user $AU_j$’s online address $Addr_j$.

2. The user $AU_j$ encrypts the on-chain address $Addr_i$ of the user $AU_i$ with his own SM4 key $K_{c_i}^j$, $SM4 : \{ Addr_j, K_{c_i}^j \} \rightarrow CT$ signs the ciphertext $CT$ with $P_{c_i}^j$ to obtain the signature value $(r, s)$, packages the authority grant instruction Command, the ciphertext $CT$, and the signature value $(r, s)$ into a data packet $\{ CT, Command, (r, s) \}$, and transmits the authority grant request to the blockchain control module of AuthROS.

3. The control module performs identity and signature verification on the received data. The ciphertext $CT$ is decoded after verification, and the SM4 key $K_{c_i}^j$ of user $AU_j$ is allocated to the chain identity $EU_j$ of user $AU_i$ and saved in the structure of the sharer key $EU_j'v$.

4. User $AU_j$ requests the shared data of user $AU_i$ using user $AU_i$’s access key.

In the preceding steps, no third party is involved, and non-essential users are unaware of the authority-granting procedure. Both parties have respect for authority. Even if the attacker registers his identity with AuthROS, he cannot rely on his identity. Consequently, no user will divulge his key to the adversary, and the adversary will be unable to access the data shared by other network users. Since the attacker cannot obtain the addresses of other network users, he cannot grant permission to share his key with other users. The network has no attacker key for the on-chain user’s identity $EU_j,v$. Therefore, no user can access the shared data, and the stolen data loses value.

Theorem 6.3. Data preservation. AuthROS can efficiently ensure the security of data transfer and promptly detect data manipulation.

Proof. AuthROS employs SM4 symmetric encryption to protect data transmission and SM2-based key exchange protocol to protect users’ encryption keys. Simultaneously, an SM2 digital signature and a data digest interaction mechanism are implemented, as AuthROS can effectively ensure the integrity and authenticity of the data. Dynamic SM3 hash will be used to verify the integrity and authenticity of data in the key interaction links, enabling the timely detection of altered data.

7. Implementation and Evaluation

This subsection introduces the hardware platform and smart contracts utilized in the experiments. Furthermore, we analyze AuthROS from response time in different perspectives, including Difficulty Settings, Message Size, Consensus Algorithms, and the Efficiency of the SM Algorithm Family. At the end of the paper, we will also discuss the security performance of AuthROS. Based on four ROS-Melodic robots, a ROS Master, and a private Ethereum network on the host, we evaluate the response time of data uploading. For identical configurations of robots, we evaluate only a single robot’s performance.

7.1. Smart Contract

The smart contracts used in experiments were written in Solidity v7.6. The contracts allowed for identity registration, Knowledge-upload, and Authority-grant, etc. All functions are listed as follows, (1) register (bytes), (2) dataUpload (bytes, bytes, bytes), (3) authorityGrant (address) and (4) dataQuery (bytes, address).

1. $\text{register (bytes)}$. This function accepts a parameter of Bytes-type, and robot owners use their SM4 keys, which are used for data encryption as a token, converted to Bytes-type, to call this method to register an identity in the Ethereum network.

2. $\text{dataUpload (bytes, bytes, bytes)}$. 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 is called. The robot owner uploads confidential data to the Ethereum network for immutable persistent storage by calling this method.

3. $\text{authorityGrant (address)}$. 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. The robot that calls this method will append the token (SM4 key). It indicates its identity to the token list in the specified EOA account so that the specified account can access the function caller’s data.

4. $\text{dataQuery (bytes, address)}$. This method accepts parameters of Bytes-type and address. The first parameter is a Bytes-type token that indicates the identity of the method caller, and the second parameter indicates the target EOA address for the query operation. The token of the method caller must exist in the token list of the target EOA so that the function caller can have access to the data shared by the target EOA.

7.2. Simulation Experiment

In some scenarios with high requirements for security performance, such as the investigation and collection of evidence at a crime scene, the images were taken by various robots at the crime scene must be filed and searchable by authorized users. Against this background, we will conduct simulation experiments; the procedure is depicted in Fig. 9.

Firstly, we start the robot and load the ROS master on the PC. After all ROS nodes within the robot have been initialized, the robot will connect to the ROS master automatically based on the configuration file. We obtain the required
Figure 9: The Process of Images Sharing through AuthROS Between Robots. Images captured by robots is converted into matrix firstly. Then the matrix will be encrypted by SM4 encryption algorithm, signed by SM2 signature algorithm and transmitted to the cloud server for persistence storage. Query of images is realized through the process of SM3 hash and ciphertext check.

facial data from the topic '/robot/CompressedImage' in the form of a 58 KB image and then matrix it using the OpenCV toolkit. Afterward, a character string will be generated from the matrix and encrypted using SM4 algorithms. The ciphertext will be signed with the user’s SM2 public key before being transmitted to the AuthROS server’s cache. Simultaneously, the SM3 cipher hash algorithm is used to generate the abstract of ciphertext, and the Data Upload interface within the smart contract is invoked to send a transaction to the Ethereum network with the abstract as a parameter. The block information is shown in Fig. 10.

Finally, the authorized robot owner can call the interface Data_Check to check the ciphertext. SM3 hashes the ciphertext within the cache, and the generated hash value is compared with the one queried from Ethereum. If the two abstracts are the same, the ciphertext in Redis, an in-memory storage structure, will be returned to the user. Otherwise, an error will be returned. The untamperability of data can be ensured in this way. At the same time, once a robot is authorized, 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 toolkit can also be used to restore the image. The whole process can be seen in the video.

(1) Consensus algorithms. To evaluate the effect of consensus algorithms, we set 300, 500, and 700 analog processes to send Data_upload requests to the Ethereum network based on PoW and PoA consensus, respectively. In addition, we evaluate the total transaction response time and success rate under three distinct concurrencies. Both networks share the same block difficulty of 0x4cccc8 and the gasLimit of 0xffffffff and the message size is 1 KB. Fig. 11 display the experimental results.

AuthROS maintains good interaction no matter which kind of consensus algorithm is used. 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 can conclude that PoA is more suitable for data transmission in multi-robots.

(2) Message sizes. In theory, when robots interact with the contract, the size and the type of the parameters passed to the contract method will impact the response time. Therefore, we conduct related experiments to study the effect of message size on blockchain networks based on two different consensus algorithms. The message size is set to 1KB, 2KB,
A Secure Data Sharing Framework for Robot Operating Systems Leveraging Ethereum

Figure 11: Different Amounts of Processes of Total Response Time and Success Rate of Transactions in Seconds for PoW-based and PoA-based Networks. Orange histogram shows PoW-based network response time, while yellow shows PoA. The red dotted dash represents PoW-based network transaction success, while the green represents PoA.

Figure 12: Average Time Consumption for Different Sizes of Messages in Seconds for PoW- and PoA-based Networks. Histograms in blue and orange represent PoW and PoA Ethereum networks, respectively.

4KB, and 8KB, respectively. We call the Data_upload interface 300 times through the simulated process and record the average time in Fig. 12.

The Fig. 12 demonstrates that the response time is proportionate to the message size, regardless of whether the Ethereum network is based on PoW or PoA consensus. However, the average response time of the PoA-based Ethereum network is significantly lower than that of the PoW-based Ethereum network. In addition, as the message size grows, the average response time of the PoA-based Ethereum network increases less than the PoW-based Ethereum network.

(3) Different settings of difficulties. We compare the effect of block difficulty on AuthROS in two consensus-based networks, including PoW and PoA. Then, we test 300 times the average time consumption of Data upload interface calls. In this case, message size is restricted to 1 KB using four block difficulty settings described in Section 4.4. Fig. 13 display the experimental results.

When PoA is used for consensus, the average time consumption does not increase with the blocks’ difficulty; it remains around 0.016 seconds. In contrast, when PoW is used for consensus, the average time consumption does not change significantly as the block difficulty increases from 0x1 to 0x10000, remaining at approximately 0.024 seconds. However, when the difficulty was increased to 0x1000000, the time required doubled. In the multi-robot interaction scenario, the information exchange of the PoA-based blockchain network is more stable and faster.

(4) Efficiency and stability of SM algorithms. In terms of data communication, AuthROS is equipped with key exchange based on SM2 to ensure the security of the SM4 key. Client-to-server communications are encrypted with the SM4 algorithm, and the ciphertext-type results are stored to ensure data privacy. Meanwhile, SM3 is used to generate the hash value of data in big size. Thus, the efficiency and stability of SM algorithms have a huge impact on the availability and speed of AuthROS.

Since the SM2 algorithm is only used in the key exchange process, accounting for a small part of the encryption process, we conduct experiments on the encryption and decryp...
Due to the characteristics of the symmetric encryption algorithm, SM4’s encryption and decryption speeds are comparable, and it is evident that the time required for decryption and encryption increases with the size of the data. Nonetheless, Fig. 14-16 demonstrates that decryption and encryption of SM4 are stable regardless of the size of the data, with encryption time consumption concentrated within a certain range. The same applies to SM3. From Fig. 15, the AuthROS equips the SM3 algorithm is excellent, completing generating 800KB data digest within the range of 6.19ms and 6.34ms.

Figure 16: 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.
8. Conclusion

This paper proposes AuthROS, the first Ethereum-based secure data interaction framework for robots with ROS. AuthROS employs the SM algorithms and data sharing protocol. The SM algorithms function as a key exchange system, with the SM2 algorithm ensuring the exchange of keys between users and the SM4 algorithm performing cryptographic operations on data. The data sharing protocol serves as a framework for providing authority to assure the reliability of the shared data and the controllability of the information data. Extensive experiments demonstrate that AuthROS can perform data exchange procedures reliably. In addition, it employs the SM3 method to process huge data batches efficiently. Through security analysis, AuthROS can also effectively resist related attacks (e.g., Node Forging).

References

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

Adam, S., Felix, L., Jenkins, X., Aleksandri, S., Joseph, C., Péter, S., et al., 2022. Go ethereum documentation. URL: https://g.ethereum.org/docs, (accessed 27 July 2022).

Alenezi, M.N., Alabdulrazzaq, H., Mohammad, N.Q., 2020. Symmetric encryption algorithms: Review and evaluation study. International Journal of Communication Networks and Information Security, 12, 256–272.

Alsamhi, S.H., Lee, B., 2020. Blockchain-empowered multi-robot collaboration to fight covid-19 and future pandemics. IEEE Access, 9, 44173–44197.

Caiazz, G., 2021. Application-level security for robotic networks. Università Ca’Foscari Venezia.

Cheminod, M., Durante, L., Valenzano, A., 2012. Review of security issues in industrial networks. IEEE Transactions on Industrial Informatics, 9, 277–293.

Drung, D., Naedele, M., Von Hoff, T.P., Crevatin, M., 2005. Security for industrial communication systems. Proceedings of the IEEE, 93, 1152–1177.

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

Ferrer, E.C., Rudovic, O., Hardjono, T., Pentland, A., 2018. Robochain: A secure data-sharing framework for human-robot interaction. ArXiv Preprint ArXiv:1802.04480.

Gupta, R., Kumari, A., Tanwar, S., Kumar, N., 2020. Blockchain-envisioned softwarized multi-swarming uavs to tackle covid-19 situations. IEEE Network, 35, 160–167.

Jiang, Y., Shang, T., Liu, J., 2021. Sm algorithms-based encryption scheme for large genomic data files. Digital Communications and Networks, 7, 543–550.

Jianhua, C., Yuefei, Z., Dingfeng, Y., Lei, H., Dingyi, P., Peng, G., et al., 2012. Public key cryptographic algorithm sm2 based on elliptic curves. URL: http://www.gmbz.org.cn/upload/2018-07-24/1532401673367034870.pdf, (accessed 1 August 2022).

Knapp, A.W., 1992. Elliptic curves. Princeton University Press.

Lamport, L., Shostak, R., Pease, M., 2019. The byzantine generals problem, in: Concurrency: the works of Leslie lamport, pp. 203–226.

Li, X., Jiang, P., Chen, T., Luo, X., Wen, Q., 2020. A survey on the security of blockchain systems. Future Generation Computer Systems, 107, 841–853.

Lopes, V., Alexandre, L.A., Pereira, N., 2019. Controlling robots using artificial intelligence and a consortium blockchain. ArXiv Preprint ArXiv:1903.00660.

McClean, J., Stull, C., Farrar, C., Mascarenas, D., 2013. A preliminary cyber-physical security assessment of the robot operating system (ros),

in: Proceedings of the Unmanned Systems Technology xv (SPIE), pp. 341–348.

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

Nishida, Y., Kaneko, K., Sharma, S., Sakurai, K., 2018. Suppressing chain size of blockchain-based information sharing for swarm robotic systems, in: Proceedings of the Sixth International Symposium on Computing and Networking Workshops (CANDARW), pp. 524–528.

Pacheco, A., Strobel, V., Dorigo, M., 2020. A blockchain-controlled physical robot swarm communicating via an ad-hoc network, in: Proceedings of the International Conference on Swarm Intelligence (ANTS), pp. 3–15.

POA, N., 2017. Proof of authority: consensus model with identity at stake. URL: https://medium.com/poa-network/proof-of-authority-consensus-model-with-identity-at-stake-d5bd15463256, (accessed 21 July 2022).

Puang, E.Y., Lehner, P., Marton, Z.C., Durner, M., Triebel, R., Albu-Schäffer, A., 2019. Visual repetition sampling for robot manipulation planning, in: Proceedings of the International Conference on Robotics and Automation (ICRA), pp. 9236–9242.

Queralt, J.P., Westerlund, T., 2019. Blockchain-powered collaboration in heterogeneous swarms of robots. ArXiv Preprint ArXiv:1912.01711.

Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., Ng, A.Y., et al., 2009. Ros: An open-source robot operating system, in: Proceedings of the ICRA workshop on open source software (ICRA), p. 5.

Quigley, M., Gerkey, B., Smart, W.D., 2015. Programming Robots with ROS: A practical introduction to the Robot Operating System. O’Reilly Media, Inc.

Roy, S., Vo, T., Hernandez, S., Lehrmann, A., Ali, A., Kalafatis, S., 2022. IoT security and computation management on a multi-robot system for rescue operations based on a cloud framework. Sensors, 22, 5569.

Sami, S., Dai, Y., Tan, S.R.X., Roy, N., Han, J., 2020. Spying with your robot vacuum cleaner: eavesdropping via lidar sensors, in: Proceedings of the 18th Conference on Embedded Networked Sensor Systems (SENSYS), pp. 354–367.

ŞEN, M.O., OKUMUS, F., KOCAMAZ, F., 2022. Application of blockchain powered mobile robots in healthcare: Use cases, research challenges and future trends. Turk Doğan ve Fen Dergisi, 11, 27–35.

Shuwan, L., Dawei, L., Chao, Z., Zhong, Z., Fang, D., Yingying, M., et al., 2012. Sm4 block cipher algorithm. URL: http://www.gmbz.org.cn/upload/2018-04-44/1522788848733658561.pdf, (accessed 3 August 2022).

Singh, P.K., Singh, R., Nandi, S.K., Ghafoor, K.Z., Rawat, D.B., Nandi, S., 2020. An efficient blockchain-based approach for cooperative decision making in swarm robotics. Internet Technology Letters, 3, e140.

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

Wood, G., et al., 2014. Ethereum: A secure decentralised generalised transaction ledger. Ethereum Project Yellow Paper, 151, 1–32.

Zhang, S., Tang, M., Li, X., Liu, B., Zhang, B., Cheng, J., et al., 2022. ROS-ethereum: A convenient tool to bridge ros and blockchain (ethereum). Security and Communication Networks, 2022.