AuthROS: A Secure and Convenient Framework for Data Sharing among Robot Operating Systems (ROS)

Shenhui Zhang[1, 2], Songjing Tao[1, 2], Shuang Wu[1], Ming Tang[1, 2], Zeping Tang[1], Zhixuan Liang[4] and Bernie Liu[2, 3 *]

[1]School of Computer Science and Technology, Hainan Univ., Haikou, 570228, China

[2]RobAI-Lab, Hainan University, Haikou, 570228, China

[3]The Hong Kong University of Science and Technology, Hong Kong

[4]Imperial College London, United Kingdom

[*]Corresponding Author: Bernie Liu. Email: by.liu@ieee.org

Abstract: Robot Operating System (ROS) has brought great potential for automation in all fields involved in production activities, which greatly improves productivity and simplifies operations that belonged to humans originally. However, ROS is highly dependent on communication but lacks secure data sharing mechanisms. Therefore, when it comes to multi-robots interaction scenarios, there are severe challenges in ensuring the safe and trustworthy exchange of specified confidential data among multi-robots. The reliability of data also needs to be considered carefully. This paper proposes a solution to this problem based on blockchain technology. We present a general secure and convenient authorization framework named AuthROS for ROS nodes with absolute security and high availability based on private Ethereum network and SM algorithms. This framework is capable of meeting the requirements for immutability and security of confidential data exchange in nodes of ROS of any size. An authority granting and identity identifying mechanism is proposed and integrated into the Ethereum smart contract to execute atomically to ensure the trustworthiness exchange of data and no third-party participation. At the same time, a key exchange based on SM2 and plaintext encryption mechanism based on SM4 is proposed to ensure the data transmission security. Furthermore, a data digest uploading scheme is adopted to improve the efficiency of querying and data uploading to the Ethereum network. Based on all of mechanisms mentioned above, AuthROS can play a role in the scenario where data transmitted among robots need to be recorded and maintain immobility. Experimental results demonstrated that AuthROS is equipped with excellent security, good time performance along with Nodes Forging resistance. We believe that AuthROS is the first secure data sharing framework for Robots loaded with ROS. We open sourced AuthROS at https://github.com/RobAI-Lab/ROS-Chain.

Keywords: ROS, Blockchain, Encryption Algorithms

1 Introduction

The Robot Operating System (ROS) [1] is an open-source meta-operating system for robots, and a distributed multi-process framework based on message passing communication, which is designed to improve the code reuse rate in the field of robot research and development. ROS provides functions similar to those provided by operating systems, such as hardware abstraction description, low-level driver management, etc. The essence of ROS is a

TCP/IP-based Socket communication mechanism [2], which can perform 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.

It is precisely because of this framework of ROS that it has some obvious drawbacks [3-5]. First of all, ROS lacks measures to ensure data security. The mode of Publisher-Subscriber makes it easy for attackers to steal data from ROS which endangers the immobility and safety of data. In addition, ROS also has problems in data reliability. Data passed among ROS nodes based on this framework can easily be intercepted or forged [25]. In the field of autonomous driving, such a situation would mean 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.

Blockchain is a securely shared decentralized data ledger, designed to facilitate the process of recording transactions and tracking assets in business networks. To begin with, the blockchain utilizes elliptic curves and hashing algorithms to generate wallet addresses, utilizes asymmetric encryption algorithms to verify transactions. Additionally, it also equips data structures such as Merkle trees to ensure data integrity and consensus mechanisms such as PoW [7] and PoA [8] to ensure data consistency. Therefore, blockchain has the advantages of decentralization, openness, autonomy, anonymity, and immutability of information. Ethereum [6] goes a step further on this basis and provides smart contract functions. It is a program running on the blockchain, and the Ethereum Virtual Machine (EVM) provided by Ethereum executes the bytecode of smart contracts. A smart contract is also an Ethereum account, which is called a contract account. It can receive and send Ether and can also save data. In this way, the contract has a state and can implement more complex logic. Therefore, to solve these drawbacks due to the ROS framework, we utilize the blockchain [6] to achieve the goals.

In addition, in order to ensure the security of the entire blockchain network and data, it is also significant to choose a series of remarkable cryptographic algorithms. The SM algorithm family is a series of algorithms formulated by the State Cryptography Administration of China. At present, three types of algorithms are mainly applied: SM2, SM3, and SM4. SM2 [9] is an elliptic curve public key encryption algorithm, including SM2-1 elliptic curve digital signature algorithm, SM2-2 elliptic curve key exchange protocol, and SM2-3 elliptic curve public key encryption algorithm. Compared with the traditional RSA algorithm, SM2 has higher encryption strength, stronger security performance and faster transmission speed. The SM3 hash algorithm is applied to the generation and verification of digital signatures. The SM3 hash algorithm is filled and iteratively compressed to generate a hash value. The hash value has a length of 256 bits, which has high security. SM4 [10] is a block symmetric cryptographic algorithm based on an unbalanced festival network, which can resist existing attacks such as differential attacks and linear attacks, and is very efficient and secure. Therefore, we can utilize the SM2 algorithm to authenticate the authority, and utilize the SM4 algorithm to encrypt the transmission and storage of confidential data.

Based on the SM algorithm family and blockchain technology, we can finally solve the existing defects of ROS. This paper proposes an efficient and secure data sharing framework

for ROS based on blockchain technology, Ethereum, Web3 and SM algorithm families: AuthROS allows critical data captured by robots shared by other robots with authority within the same Ethereum network.

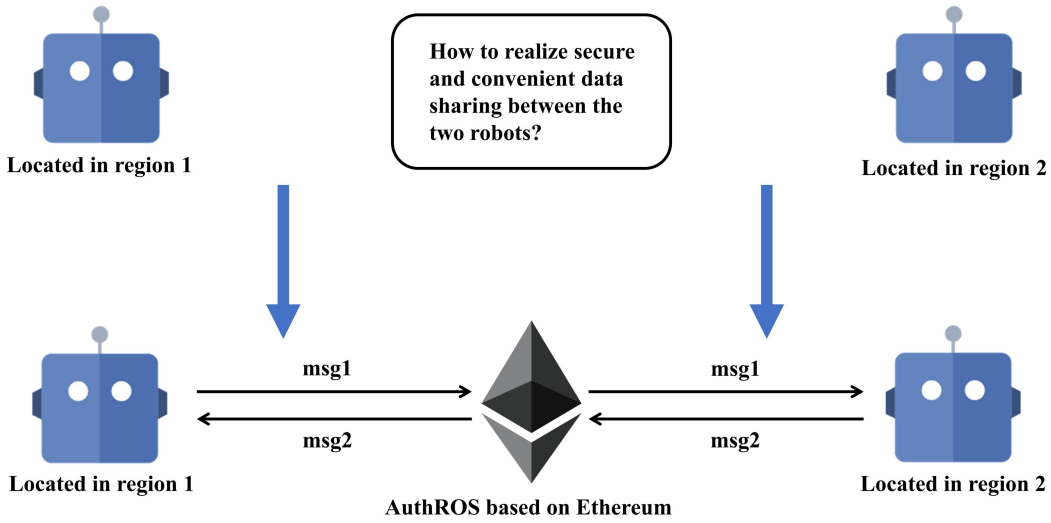

Fig 1. There exist a lot of attack means against the current data sharing in ROS like Nodes Forging, Eavesdropping and repetition etc. To solve or reduce the risk brought by these attack means as much as possible, we propose a data sharing framework AuthROS

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 family can effectively ensure the security of data transmission. The superiorities of AuthROS are shown below:

1. AuthROS provides user-friendly interfaces, hiding the complicated implementation of blockchain and the details of the encryption algorithm. The users are just required to input the names of topics or nodes to be monitored to interact with Ethereum.

2. AuthROS has high safety performance. AuthROS introduces a symmetric key plain text encryption mechanism, providing a shield for the data against the third party. An authority granting mechanism is introduced to keep the robots' data safe. Only when the robots with the data owner's permit can these robots access the data.

The main contributions of the paper are:

1. Proposing an efficient and secure general confidential data sharing framework for ROS based on blockchain technology.

2. Presenting an authority granting mechanism to control the access to the specified confidential data transmitted among ROS based on Ethereum.

3. We realized a kind of encryption communication of ROS based on SM algorithm.

4. We developed a secure, reliable and convenient Ethereum interaction solution for ROS-based robots.

The rest of this paper is organized as follows: section II describes the related works, section III formalizes our methodology, section IV introduces the framework of AuthROS

and section V analyzes the results of experiments. The end is a summary.

2 Related Works

In the present national and international study, with the increasing research on robotic swarm communication and the applications of blockchain technology, the latency of them was measured and analyzed. Besides, researchers have also identified parts of links between them. Most of these studies focus on the blockchain-based communication within robotic swarm, secure information sharing through blockchain network and settlement of Byzantine problem [11].

Some research focusing on swarm communication are listed as follow. Ming Li et al. described the possible ways of building wireless mesh networks to enable multiple robots within every cluster to form mobile self-organizing networks [12]. John Harris et al. proposed an approach working on a generic robotic swarm communication network framework through scalable algorithms for autonomous cluster deployment and ROBOTRAK (a socket-based cluster monitoring and control) [13]. Eduardo Castelló Ferrer et al. introduced the first learning framework for secure, decentralized and computationally efficient data and model sharing among multiple robot units installed at multiple sites [14]. Pramod et al. used a set of experiments to validate that the Ethereum can be a secure media for communication for multiple small Unmanned Aerial Vehicles (sUAVs) [15].

Research focus on secure information sharing also weighs a lot. To combat COVID-19 pandemic, a conceptual, trustworthy framework which can facilitate the sharing of information, goals and representation was proposed by Alsamhi et al. To increase the intelligence, decentralization and autonomous operations of connected multi-robot collaboration in the blockchain network [16]. Meanwhile, in the case of mutual collaboration in edge-assisted multi-robot systems, Jianan Li et al. adapt alike ideas and proposed a blockchain-based collaborative edge data inference (BCEI) framework for edge-assisted multi-robot systems[17]. The framework of data sharing of Internet of Things (IoT) platform with the help of Ethereum Blockchain was presented by Xia Qi [18]. Ilya Afanasyev et al. firstly presented the analysis and application of blockchain-based multi-intelligent body robotic systems [19].

There comes with the classic Byzantine problem within Robotic swarms. To neutralize Byzantine Robot within swarm, Jianan Li et al. propose a blockchain-based collaborative edge data inference (BCEI) framework for edge-assisted multi-robot systems [20]. Alexandre Pacheco et al. presented that robotic clusters used decentralized mobile self-organizing networks to achieve a high-throughput communication framework and use blockchain as a security layer to neutralize Byzantine robots [21]. Volker Strobel et al. demonstrated how robotic swarm achieve consensus in the presence of Byzantine robots by exploiting blockchain technology [22].

3 Key Conception

The framework of AuthROS is depicted in Fig 2. The whole framework is divided into two parts: ROS in local robots and blockchain network. The core logic of the framework is that the confidential data in the local ROS is shared in the blockchain network in the form of ciphertext. The data provider can grant access to data to other members of the blockchain

network. To give a better presentation of AuthROS, in this subsection, we will introduce the implementation of some key mechanisms in AuthROS such as identity identifying, Authority granting and data sharing, etc.

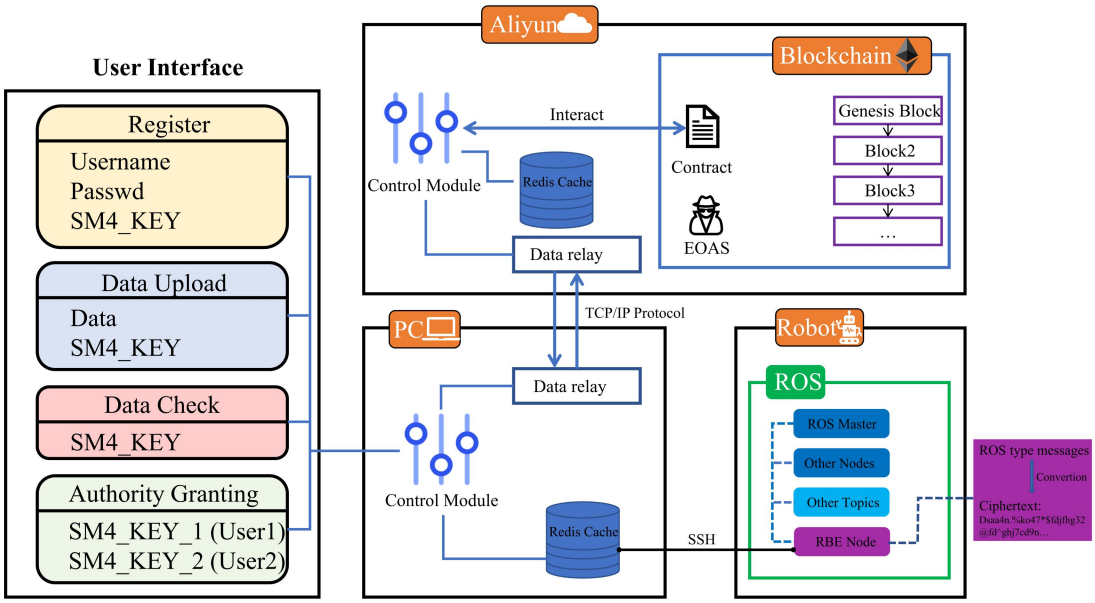

Fig 2. Framework of AuthROS

a. SM Algorithm family including SM2, SM3 and SM4

The encryption of AuthROS is based on SM2, SM3 and SM4. The SM2 elliptic curve public key crypto algorithm involves SM2-1 elliptic curve digital signature algorithm, SM2-2 elliptic curve key exchange protocol and SM2-3 elliptic curve public-key encryption algorithm which are used to implement digital signature, key negotiation and data encryption respectively [23]. The design of SM3 builds upon the design of the SHA-2 hash function, but introduces additional strengthening features in safety and encryption speed [26]. The SM4 block cipher algorithm is an iterative block cipher algorithm, which consists of an encryption and decryption algorithm and a key expansion algorithm. The SM4 block cipher algorithm adopts an unbalanced structure, the block length is 128b, and the key length is 128b. Both the encryption algorithm and the key expansion algorithm use a 32-round non-linear iterative structure. The algorithm structure of the encryption operation and the decryption operation are the same, and the round key of the decryption operation is used in the opposite order to that of the encryption operation [24]. For implementation of the algorithms, please refer to [23, 24].

b. Key Exchange mechanism based on SM2

To guarantee the security of SM4 key which is used to encrypt plaintext to the most, SM2 algorithm is used to bring about a mechanism of key exchange. When robot owner firstly use AuthROS, they are required to input their own SM4 key for future encryption of sharing data. After the SM4 key is input, the process of key exchange will be executed, immediately. The SM2 public key PB built in the framework is used to encrypt SM4 plaintext PT. After the ciphertext is transmitted to the blockchain end via TCP / IP transmission protocol, the ciphertext will be decrypted using SM2 private key PR. Obtained

SM4 key will be stored in blockchain network for further decryption works. At the same time, the robot sending ciphertext will also be tied with an Ethereum account (EOA).

c. Local ROS Topics

The core of AuthROS is a monitoring node. AuthROS can monitor the topics they want through 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 monitoring node parses the corresponding messages after obtaining them from topic named '/robot/odom', and extracts useful information like three-axis velocity and angular velocity at a given time etc. Fig 3 shows how AuthROS obtain and parse the Odometry messages.

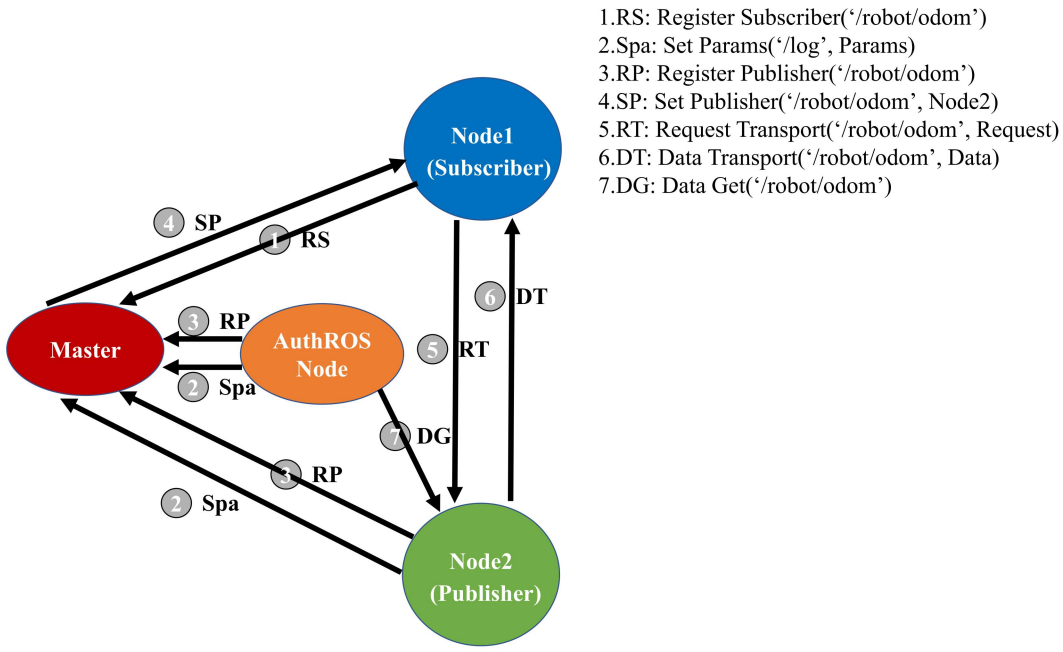

Fig 3: This is the workflow of ROS in local robots. Firstly, node1, node2 and AuthROS node should register their identity (subscriber or publisher) through ROS Master and set the topic '/robot/odom' for message transmission. Then, AuthROS node can seize data at any given time when the communication between node1 and node2 is going on. Then, extracted messages will go through a range of format conversion and encryption which will be introduced later.

d. Topics Input and Data Transmission

Through input of the names and types of wanted topics, AuthROS is able to start the monitoring process automatically. Then AuthROS will extract valuable data K from the specific raw messages. Extracted data will be converted into plaintext wrapped in JSON format J, which includes exact time of extracting $Time$, types of raw messages $Type$ and data sharer $Owner$ etc. The typical format of J is depicted in Fig 4:

```
{
    "K": "6dsa7Dd7EdasGB9fgH0J9f6Hlzds0...",
    "Owner": "ZSH",
    "Time": "2022-3-01 18:06:34",
    "Type": "Odometry",
    "Command": "Data_upload",
    "Params": {
        "address": "0x8b5414b4f8ffe4cf2ec7cafabff218df20ca71a1",
        "Key": "XXXXXX"
    }
}
```


Fig 4. Typical format of wrapped data in JSON

Next, the data in JSON format will be encrypted by the SM4 key entered in step 1 and transmitted to the blockchain for parsing through TCP / IP protocol. Finally, the ciphertext will be decrypted by the SM4 key stored in the blockchain network and going through the process of interaction with blockchain according to the Command *Command* .

e. Identity Identifying resisting Nodes Forging

There are several ways that an hacker can manipulate ROS. We can conclude that the kind of attacks against ROS are common to several IoT/CPS frameworks like Man in the middle, Repetition and Injection etc from analysis from the research from Gianluca Caiazza [25]. With SM algorithm family, AuthROS is already resistant to most attacks mentioned above. However, there is an attack that can not be effectively defended by relying only on encryption. We all know that ROS allows multiple message subscribers and publishers under one topic. Then, in the local ROS mentioned in the previous section, an attacker may build a publisher with the same name as the local ROS, replacing the publisher in the local ROS to output the attacker's own information to some topics in the local ROS or pretending to be the subscriber in the local ROS to steal data. We call this attack method Nodes Forging or Nodes Impersonation. Typical Nodes Forging is shown in Fig 5.

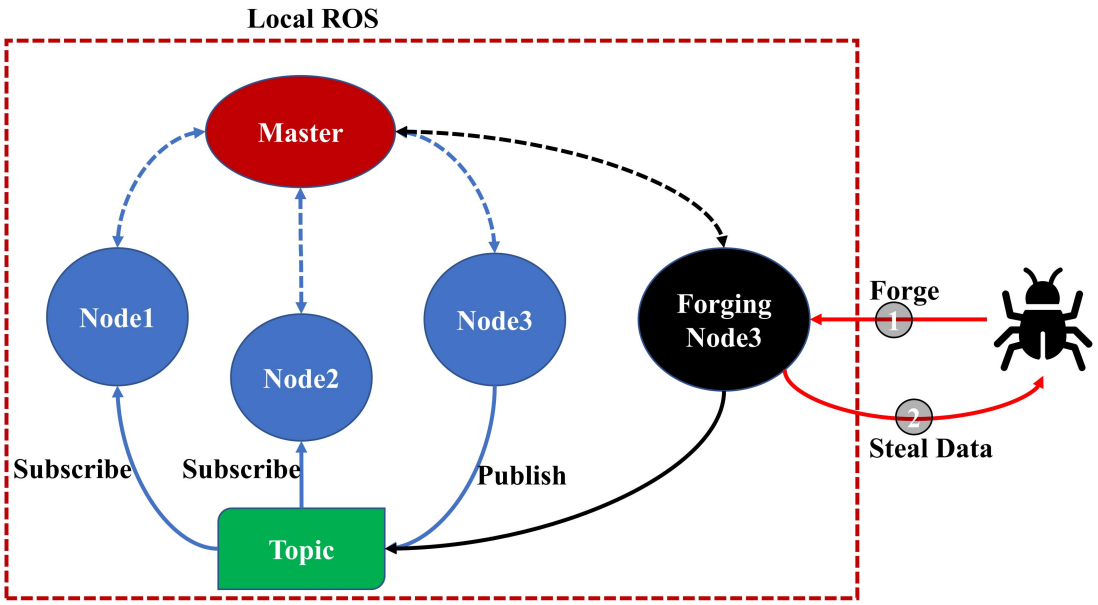

Fig 5. Attacker implant the forging node into the local ROS, using the same name as the node running in the local ROS, to steal data transmitted among the nodes running in the local ROS or disrupt the local ROS.

In the scenario of multi-robot interaction, Nodes Forging poses an incalculable risk. Attackers can use their own nodes to implant information into the robot swarm, disrupting or paralyzing the normal work of the swarm. To minimize this risk, we designed a two-layer Identity identifying mechanism for AuthROS. Main working principles is shown in Fig 6. Firstly, robot owner must register in the system with their username, password and SM4 key. Then can the robot owner log in to the system with the set of username and password. The SM4 key will be bind with an EOA while the pair of username and password will be stored in the Redis. This is the first level of Identity identifying. Secondly, when the robot owner wants to call other interfaces loaded in the contract, AuthROS will check there is a corresponding key under the robot's Ethereum account matches the key transferred by the user through the **Identity Check** mechanism. This is the second level of Identity identifying. The two-layer Identity identifying mechanism can effectively resist Forging attacks. This will be confirmed in the following experiments.

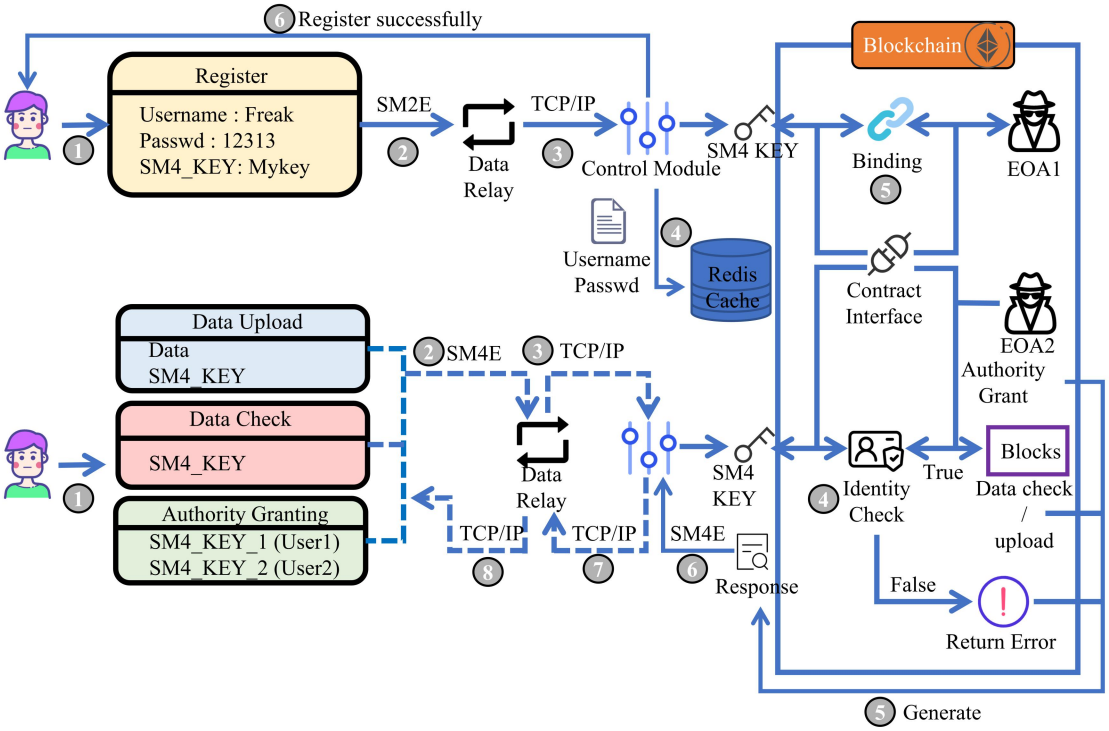

Fig 6. Different interfaces are provided for AuthROS users to use AuthROS. Robot owners are required to register in AuthROS through a pair of username and password along with a SM4 key. Then can they use AuthROS through interaction with the functions loaded in the contract. Every request for interaction must be tied with a SM4 key to ensure the right implementation of Identity Check. Only the authorized user can access the data shared by other users. User can authorize other users within the Ethereum network through the Authority_Grant interface.

f. General trustworthy Data Sharing

General trustworthy Data sharing of AuthROS is realized mainly through Authority Granting mechanism. Authority is granted through key transmission in the blockchain network. Robot owner can freely view the ciphertext of confidential data on the blockchain shared by themselves. And robot owner can call the interface on blockchain to attach the key stored in their Ethereum account to other Ethereum accounts within the network. The owner of the robot whose Ethereum account obtains the key can query the data ciphertext shared by the corresponding key owner, and decrypt the ciphertext with the key to obtain the plaintext. This Authority Granting mechanism can not only effectively ensure the security of information sharing, but also has good anonymity because the key is transmitted in the chain using the Ethereum account address. Principles of Authority Granting is depicted in Fig 7.

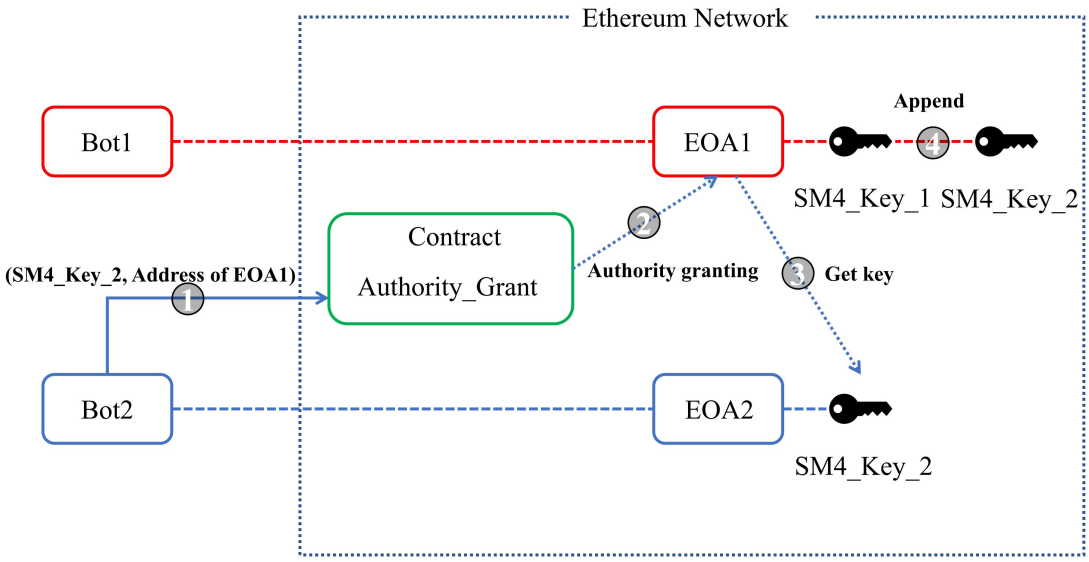

Fig 7. EOA represents Externally Owned Accounts which is introduced in section 4. Bot2 use the interface 'Authority_Grant' written in the smart contract with parameters of address of EOA1 and its own SM4 key to attach the key to the EOA1. Then EOA1, tied with Bot1, can access to the data shared by Bot2.

4 Framework of AuthROS

a. Ethereum Setup

Ethereum Virtual Machine (EVM), is a Turing complete virtual Machine that processes and handles all transactions carried out in the Ethereum. A private Ethereum network where every robot which run an independent ROS has a relating Ethereum account is set up on the server with the help of Geth. The identity of every robot is known and verifiable due to features of a private blockchain. Then, we will introduce Geth, Truffle and architecture of Ethereum network of AuthROS in the following.

1) Geth: Geth client is used to build a private Ethereum blockchain. Ethereum clients are software that can establish P2P communication channels with other robots, sign and broadcast transactions, mine, deploy, and interact with intelligent contracts. So far, Geth is one of the most popular Ethereum clients, which is written in the Golang language. There are two types of accounts in Geth: Externally Owned Accounts (EOA) and Contract Accounts (Smart Contracts). AuthROS has a built-in Contract Account managed by administrator in its private Ethereum blockchain, and the rest of the robot-bound accounts are EOA accounts. An Ethereum account is defined by a pair of keys including public keys and private keys, and each account is indexed by its address which is derived from the first 20 bytes of the SHA3 hashed public key.

2) Truffle Development Environment: Truffle is an Ethereum resource management channel, development and testing environment that makes development on Ethereum easy. Truffle can compile, link, and deploy smart contracts. Truffle can manage the binary files generated by contract compilation. Truffle employs the package management provided by EthPM & NPM and employs the ERC190 standard. Robots can directly test the contract via

console. The built-in automation script of Truffle ensures that each node can only call the methods meant for it, ensuring the security of communication between Ethereum nodes.

3) Architecture of Ethereum network: Building an Ethereum node on the robot with limited computational power will encounter the bottleneck of Edge Computing. Therefore, the Ethereum network composed of 3 nodes was implemented using a host. The robot only needs to maintain a connection with the ROS Master running on the host. Keeping the ROS separated from the Ethereum network can avoid the bottleneck of Edge Computing. The structure of the Ethereum private chain is shown in Fig 8. Among the 3 Ethereum nodes, one node is a miner node. This node was also designated as the bootstrap node and was responsible for forming the overlay network with other nodes. Four ROS-Melodic robots are associated with two EOA accounts in Node2 and Node3 respectively to obtain the ability to interact with the Ethereum network.

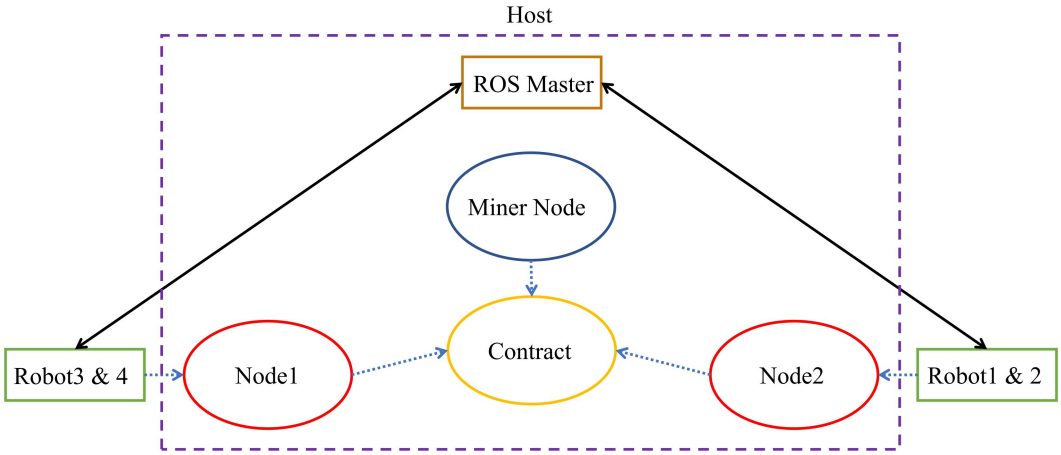

Fig 8. Framework of private Ethereum network used in experiments

b. Consensus Algorithm

Consensus algorithms equips the distributed ledging of the blockchain with immutability, automation and anonymous transactions without third parties. Blockchain can only be an inefficient distributed storage database if without consensus algorithms. The existence of the consensus algorithms endows blockchain technology with core meaning and value. Current consensus algorithms mainly include Proof of Work (Pow), Proof of Stake (PoS), Proof of Authority (PoA), and Practical Byzantine Fault Tolerance (PBFT). Ethereum currently uses PoW consensus algorithm. With ETH2.0 coming soon, consensus algorithm of Ethereum will be changed from PoW to PoS. Since Ethereum has not yet fully supported PoS, in our experiments, we evaluated the effect of PoW and PoA algorithms on time consumption of data on-chain and query. The biggest difference between PoW and PoA is that PoW relies on mining to verify blocks and cannot specify the block times, while PoA relies on pre-authorized nodes to verify blocks without mining operations. Therefore, the block time can be specified.

c. Genesis Block

Genesis block is the first block in the blockchain. Some important parameters defining the blockchain was included in the genesis block. Normally, the file which is used to initialize

genesis block is written in JSON and is shared by all Ethereum nodes. The file we used to initialize a PoW-based Ethereum network is depicted in Fig 9.

```
{
  "config": {
    "chainId": 666,
  },
  "nonce": "0x0",
  "timestamp": "0x5ddf8f3e",
  "extraData": "0x0",
  "gasLimit": "0xffffffff",
  "difficulty": "0x4cccc8",
  "mixHash":
"0x0000000000000000000000000000000000000000000000000000000000000000",
  "coinbase": "0x0000000000000000000000000000000000000000",
  "alloc": { },
  "number": "0x0",
  "gasUsed": "0x0",
  "parentHash": "0x0"
}
```

Fig 9. Genesis file of a PoW-based private Ethereum network

Some key parameters include:

- ChainId: This parameter is the network ID of the blockchain, which is used when local private blockchain network connect to other network nodes. The public network ID of Ethereum Main Network is 1, and network nodes with different IDs cannot connect to each other.

- GasLimit: This parameter is the maximum amount of gas that the robot of Ethereum is willing to spend for a particular transaction. While gas is a unit used to measure the effort required for a particular computation in an Ethereum Virtual Machine (EVM). Gas price is the value the transaction sender is willing to pay per gas unit and is measured in GWei. Ether is the token used to pay for gas.

- Difficulty: This parameter represents the difficulty of generating blocks in the blockchain, that is, the difficulty of calculating the hash value of the next block. The higher the difficulty is set, the longer it takes to find blocks.

- Alloc: This parameter is used to specify the number of preset accounts in the blockchain and the amount of ether allocated to these preset accounts. The specified amount of ether balance(特殊处理) cannot be lower than the preset GasLimit, otherwise there will be a situation where the amount of ether in an account is not enough to pay for the transactions it needs to execute. In addition, it is worth mentioning that the Alloc(特殊处理) parameter only exists in private blockchains.

To evaluate the effect consensus algorithms on blocking time and verification time of transactions, we also prepare a genesis file depicted in Fig 10 to initialize a PoA-based Ethereum network to make comparations with PoW-based network. And for the effect of

Difficulty on the verification time of transactions, the block difficulty values were set to the following values in both of the genesis files.

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


```
{
  "config": {
    "clique": {
      "period": 20,
      "epoch": 30000
    }
  },
  "nonce": "0x0",
  "timestamp": "0x62232e59",
  "extraData":
"0x0000000000000000000000000000000000000000000000000000000000000000d2f01c1f7
341f67105fff23aab886ce0fdff561b0000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000",
  "gasLimit": "0xffffffff",
  "difficulty": "0x4cccc8",
  "mixHash":
"0x0000000000000000000000000000000000000000000000000000000000000000",
  "coinbase": "0x0000000000000000000000000000000000000000",
  "alloc": {
    "d2f01c1f7341f67105fff23aab886ce0fdff561b": {
      "balance":
"0x2000000000000000000000000000000000000000000000000000000000000000"
    }
  },
  "number": "0x0",
  "gasUsed": "0x0",
  "parentHash":
"0x0000000000000000000000000000000000000000000000000000000000000000"
}
```


Fig 10. Genesis file of a PoA-based private Ethereum network.

Both of the PoW and PoA based private Ethereum network share the same set of **GasLimit**, **Difficulty** and **Alloc**. All EOAs will be created and allocated balances after the completion of initialization of genesis block. Meanwhile, a contract account in the miner node will be created for contract deployment. After the deployment, the contract account will turn into a miner, which provides Ethers for other EOAs. Though a private Ethereum network is

not part of the public blockchain, we set gas prices to default value to simulate the true trading situation.

5 Implementation and Experimental Results Evaluation

This subsection introduces the hardware platform and smart contracts exploited in experiments, and then analyzes AuthROS from the perspectives of response time in different situations: different settings of Difficulties, message size, consensus algorithms and efficiency of SM algorithm family. We will also discuss the safety performance of AuthROS in the end of paper. We evaluate response time of data uploading on the base of 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 one robot.

a. 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 which has a built-in 9-axis IMU sensor, RPLIDAR A1 radar, A Wi-Fi module that can provide up to 867mbps communication bandwidth and a 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 robots, as long as the personal computer running ROS Master. Through the Wi-Fi module each robot can connect to the ROS Master to realize stable communication. The personal computer running ROS Master also connect to the server hosting Ethereum network, thus can provide interaction between robots and Ethereum.

The controller and Jetson Nano were connected through a UART connection using software function calls provided by the controller's onboard C++ SDK. the controller collect 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 is with a CPU (Intel Xeon(Ice Lake) Platinum 8369B @3.5GHz), memory (16GB DDR4 3200MHz) and disk (80GB ESSD).

b. Smart Contract

The smart contracts used in experiments were written in Solidity v7.6. The contracts allowed for the identity registration, Knwoledge-upload and Authority-grant etc. Fig 11 depicts the workflow of AuthROS. All functions are listed as follow:

- **Register(bytes):** This function accepts a parameter of type Bytes and robot owners use their SM4 keys which are used for data encryption as token, converted to type Bytes, to call this method to register an identity in the Ethereum network.
- **Data_upload(bytes, bytes, bytes):** This function accepts three parameters of type Bytes, the first parameter is the data ciphertext of type Bytes 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. The robot owner

uploads confidential data to the Ethereum network for immutable persistent storage by calling this method.

- **Authority_Grant(address):** This function accepts a parameter of type address, which is the address of the EOA that will be given access to function caller's data. 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.
- **Data_query(bytes, address):** This method accepts parameters of type bytes and address. The first parameter is a byte-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.

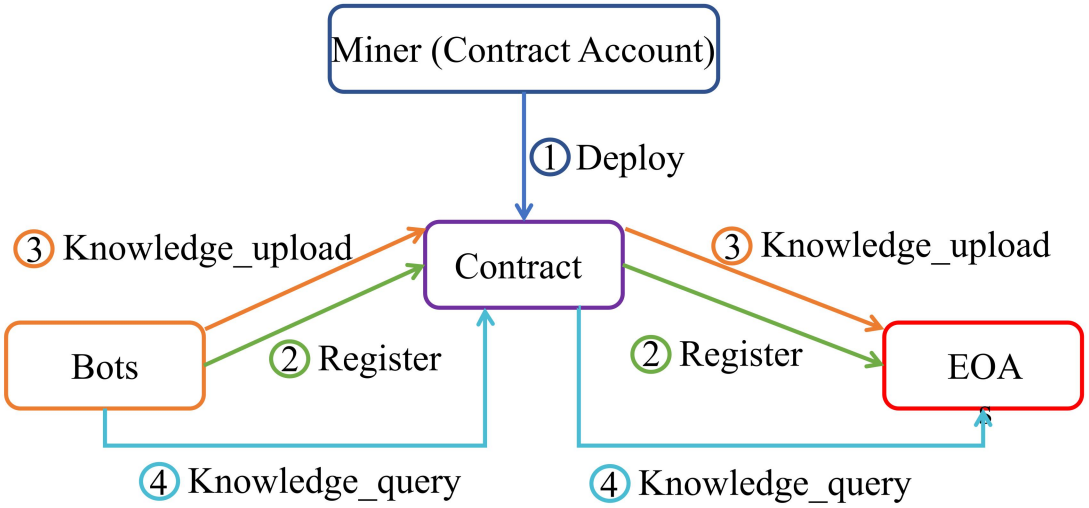

Fig 11. Workflow of our Ethereum network used in experiments. All bots are tied with a EOA and can interact with blockchain through smart contract functions. The smart contracts featured functions which can only be called by pre-set EOAs. In this case, bots firstly need to register to get the tied EOA and then can call other functions.

c. Simulation experiment

In some scenarios with high requirements for safety performance, such as in the process of investigation and evidence collection at the crime scene, the photos taken of by different robots at the crime scene needs to be filed and searchable by authorized users. We will carry out simulation experiments against this background and the process of which is shown in Fig 12.

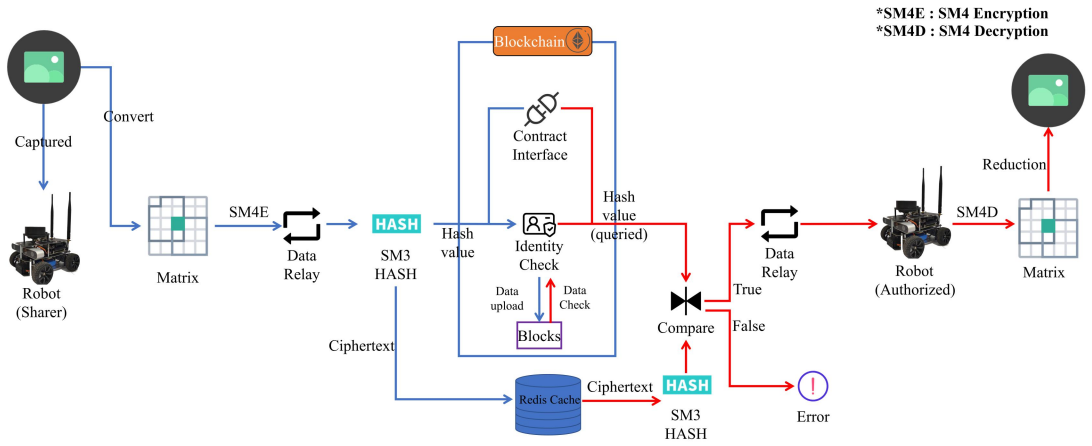

Fig 12. Process of figure sharing. Figures captured by robots is converted into matrix firstly. Then the matrix will be encrypted by SM4 and transmitted to the cloud server for persistence storage. Query of figure is realized through the process of SM3 hash and homomorphic encryption.

Firstly, we start the robot and load the ROS master on the PC. after all ROS nodes within the robot are initialized, the robot will automatically connect to the ROS master according to the configuration file. We get the facial data we need from the topic /robot/complicatedImage in the form of picture of 58 KB and then use the Opencv toolkit to matrix it. Next, the matrix will be converted into character string and encrypted by SM4 algorithms. The ciphertext will be transmitted to the Redis 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 interface Data_Upload within the smart contract will be called to send a transaction with the abstract as parameter to the Ethereum network. The block information is shown in Fig 13.

```
{
   "blockhash":
"0x1dedaa7bd88dabef8e2dc2064253ae9d03d30f37d54e1ea3dba402b3eb5a37b3",
   "blockNumber": 14
   "from": "0xcc047453d339990289bd67e8163802ca169e2f61",
   "gas": 255062,
   "gasPrice": 1000000000,
   "hash": "0xdd16c31c1169a39d9c4307f881070f853f98d8b4ee40e36165be9bddb31c32a0",
   "input":
"0xd033688b0000000000000000000000000000000000000000000000000000000060
0000000000000000000000000000000000000000000000000000000000000000c000000000000
0000000000000000000000000000000000000000000000010000000000000000000000000
0000000000000000000000000000000000000040333066356235386339353135646533633
834343238346365613037643563326634346565316165616533636566383165373334616331
633965316133303461640000000000000000000000000000000000000000000000000000000000000
0000000037a736800000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000013323032322d303
42d32332032313a34393a313200000000000000000000000000",
   "nonce": 1,
   "r": "0xe8d5db31094f066a568ceb86a2e3df95b940797d7a3a52c2a4005f7d71ed48d2",
   "s": "0x4aa58226addbaaff5f81cc7ba60434329cc3b4f0b03b9da8c3ee60e20e0e8936",
   "to": "0x81f988ee5792e8c46d8213ee64723a2858b8fa7e",
   "transactionIndex": 0,
   "v": "0x558",
   "value": "0",
}
```


Fig 13. Information of figure uploading where all related information is recorded in the block and each block is chained through the **blockhash**.

Finally, the authorized robot owner can call 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 the Ethereum. If the two abstracts are the same, the ciphertext in Redis will be returned to the user, otherwise an error will be returned. Non-tamperability of data can be ensured through this way. 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 toolkit can also be used to restore the figure. The whole process can be seen in the video.

d. Consensus Algorithm

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, test the total response time and success rate of transactions under

three different concurrency. Both networks share the same block **difficulty** of *0x4cccc8* and **gasLimit** of *0xffffffff* and the message size is **1 KB**. The experimental results are shown in Fig 14.

AuthROS maintains good interaction using both algorithms in high concurrent scenes. For success rate of transactions, both consensus algorithms reached more than 99% and it was observed that different consensus algorithm has no significant gap in success rate of transactions. However, it was obvious that PoA has a clear edge over PoW on the response time. This behavior follows the fact that PoW reles on the mining process to verify transactions while PoA verifies transactions through pre-set nodes' voting which causing longer blocking time and verifying time of transactions of PoW. We can draw a conclusion that PoA is a faster option for data transfer in multi-robots.

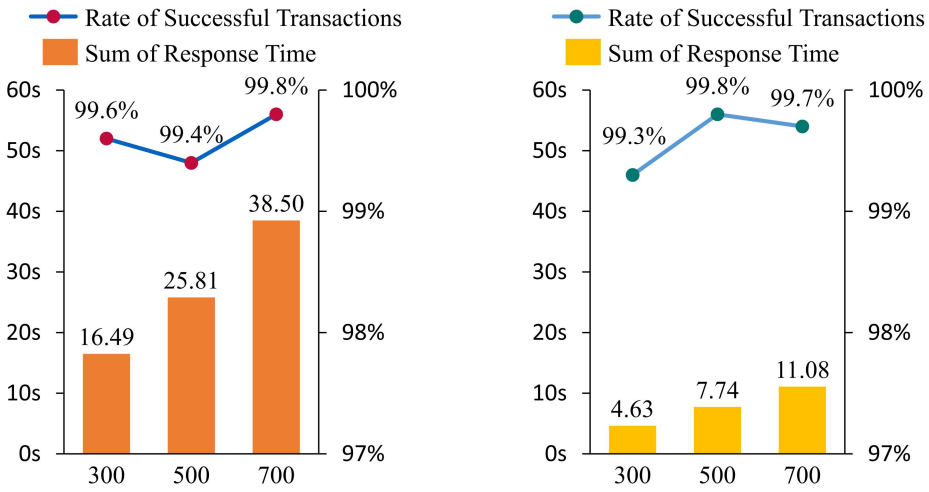

Fig 14. Left: total response time and success rate of transactions in seconds for Proof of Work (PoW). Right: total response time and success rate of transactions in seconds for Proof of Authority (PoA).

e. Messages Size

In theory, when robots interact with the contract, the size and even the type of the parameters passed to the contract method will 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**. Call the **Data_upload** interface 300 times through the simulated process and record the average time. The experimental results are shown in Fig 15.

It can be observed from the figure that whether the Ethereum network is based on PoW consensus or PoA consensus, the response time always increases with the increasement of message size. But in comparison, the overall average response time of the PoA-based Ethereum network is lower than that of the PoW-based Ethereum network. And as the message size increases, the average response time of the PoA-based Ethereum network also has a lower level of increasement than that of the PoW-based Ethereum network.

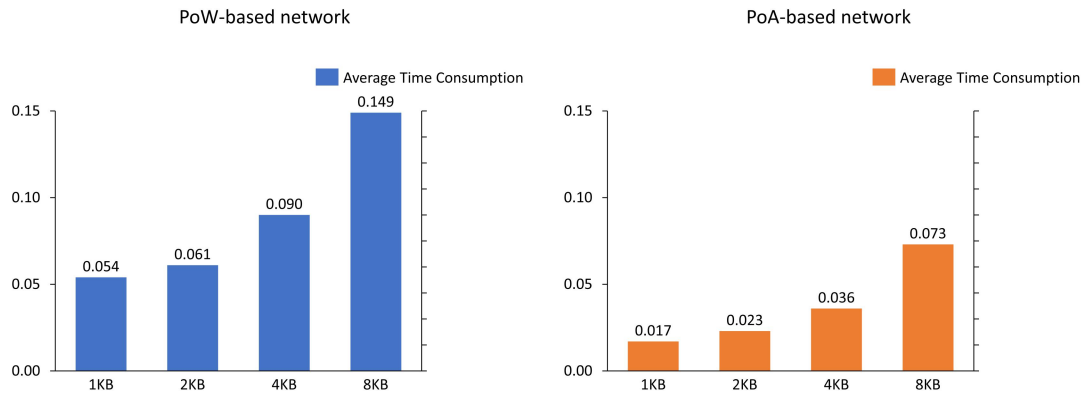

Fig 15. Left: average time consumption for different sizes of messages in seconds for Proof of Work (PoW) based network. Right: average time consumption for different sizes of messages in seconds for Proof of Authority (PoA) based network.

f. Different settings of Difficulties

In two different consensus-based networks (PoW and PoA), we use 4 different block **difficulty** settings mentioned in section 4-e to test the average time consumption of 300 **Data_upload** interface calls where message size is restricted to **1 KB** to compare the effect of block difficulty on AuthROS in different consensus-based networks. The experimental results are shown in Fig 16.

When PoA was used for consensus, the average time consumption does not increase with the increasement of block **difficulty**, remaining at around 0.016s. In contrast, when PoW was used for consensus, the average time consumption does not change significantly when the block **difficulty** gradually increases from *0x1* to *0x10000*, maintaining at around 0.024s. But, when the **difficulty** was set to *0x1000000*, time consumption doubles. It can be seen that the information exchange of the PoA-based blockchain network in the multi-robots interaction scenario is more stable and faster.

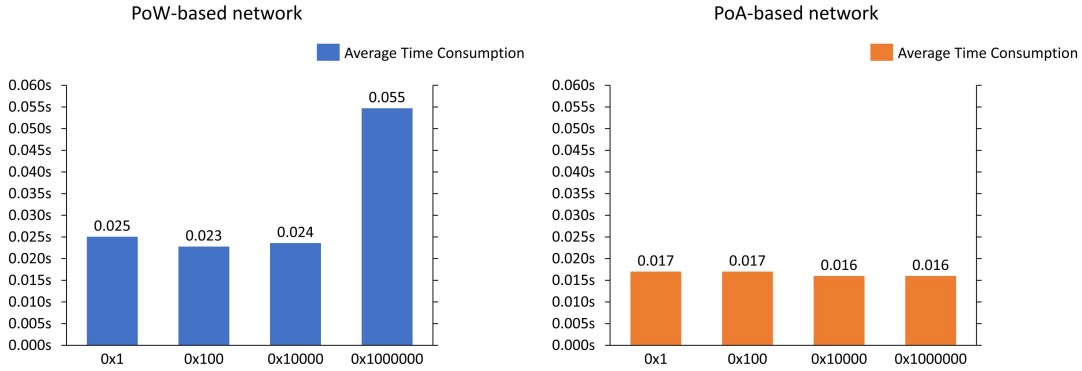

Fig 16. Left: average time consumption in seconds for Proof of Work (PoW) based network. Right: average time consumption in seconds for Proof of Authority (PoA) based network.

g. Efficiency and Stability of SM Algorithm Family

In term of data communication, AuthROS is equipped with key exchange based on SM2 to ensure security of SM4 key, encrypting the original data in communication between the

client and the server through SM4, keeping the transmission in the form of ciphertext to guarantee the privacy of data. 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 decryption speed and stability of SM4 and SM3. We use plaintext data with sizes of 1KB, 2KB, 4KB and 8KB as encrypted source data for SM4, encrypting and decrypting the data 300 times respectively and recording the average time consumption. We also evaluate the speed and stability of SM3 using a 800 KB matrix as encrypted source data for digest generation in 300 times. The experimental results are shown in Fig 17-26.

Due to the characteristics of the symmetric encryption algorithm, the encryption and decryption speeds of SM4 are close and it is clear that the time consumption of decryption and encryption increases with the increasement of data sizes. But, it is also observed from Fig 18-25 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. From Fig 26, it is clear that the stability of SM3 algorithm is excellent, floating between 6.19s and 6.34s.

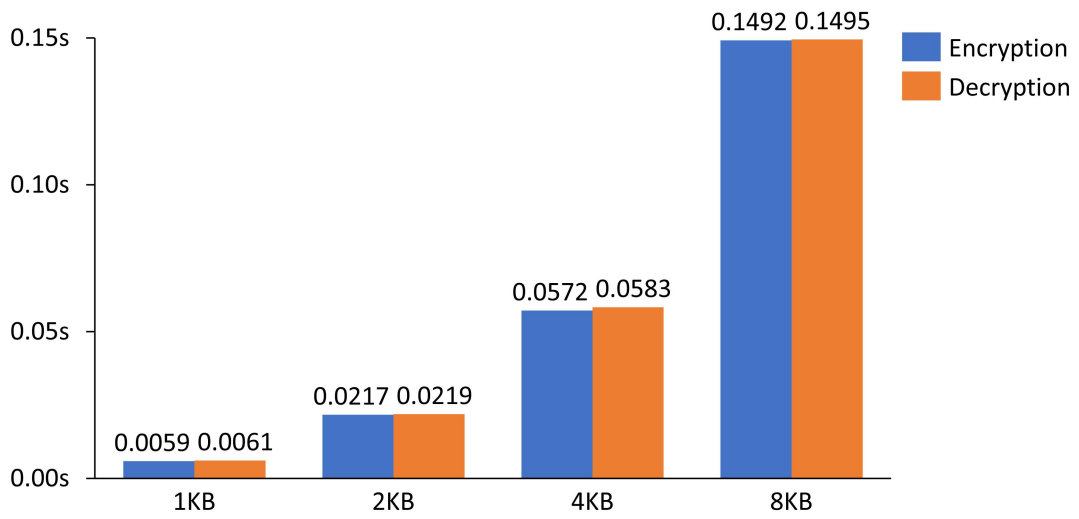

Fig 17. Histogram in blue represents the average time consumption of data encryption while histogram in orange represents the average time consumption of data decryption.

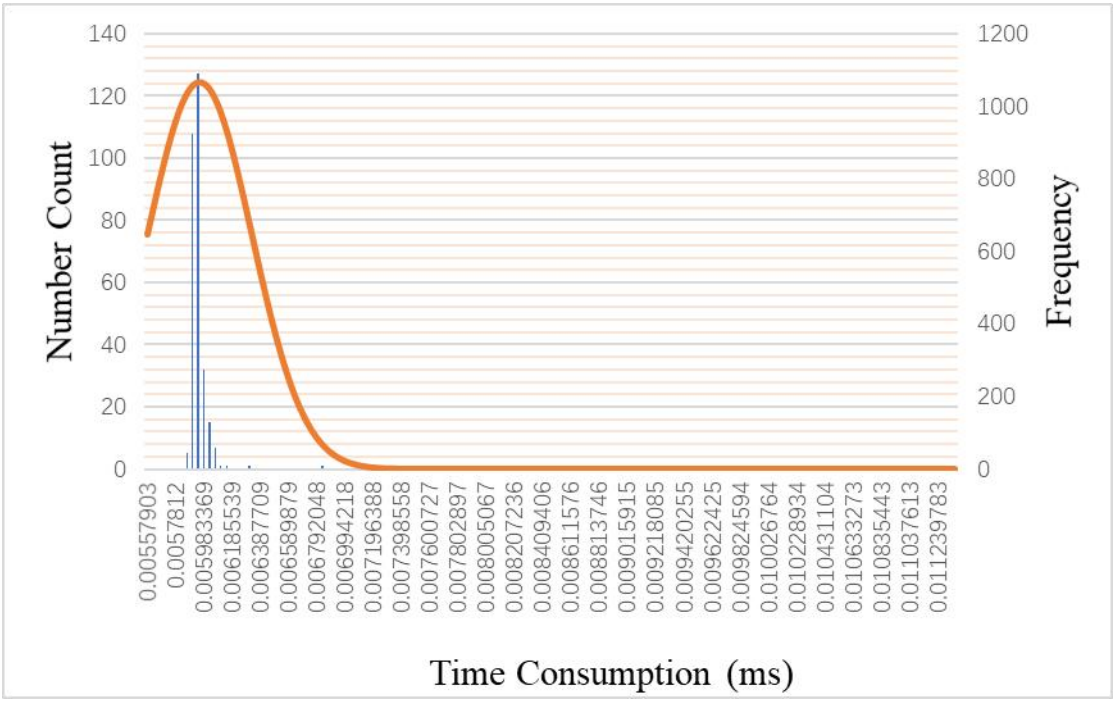

Fig 18. Time consumption of 300 data encryption each time (1 KB)

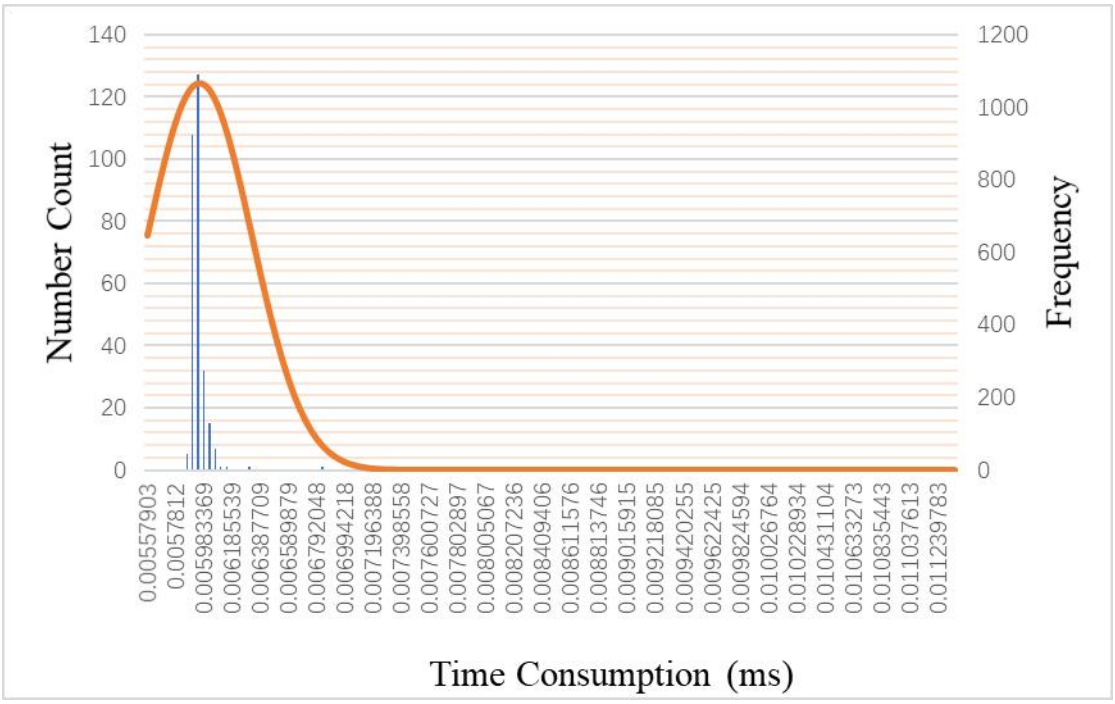

Fig 19. Time consumption of 300 data decryption each time (1 KB)

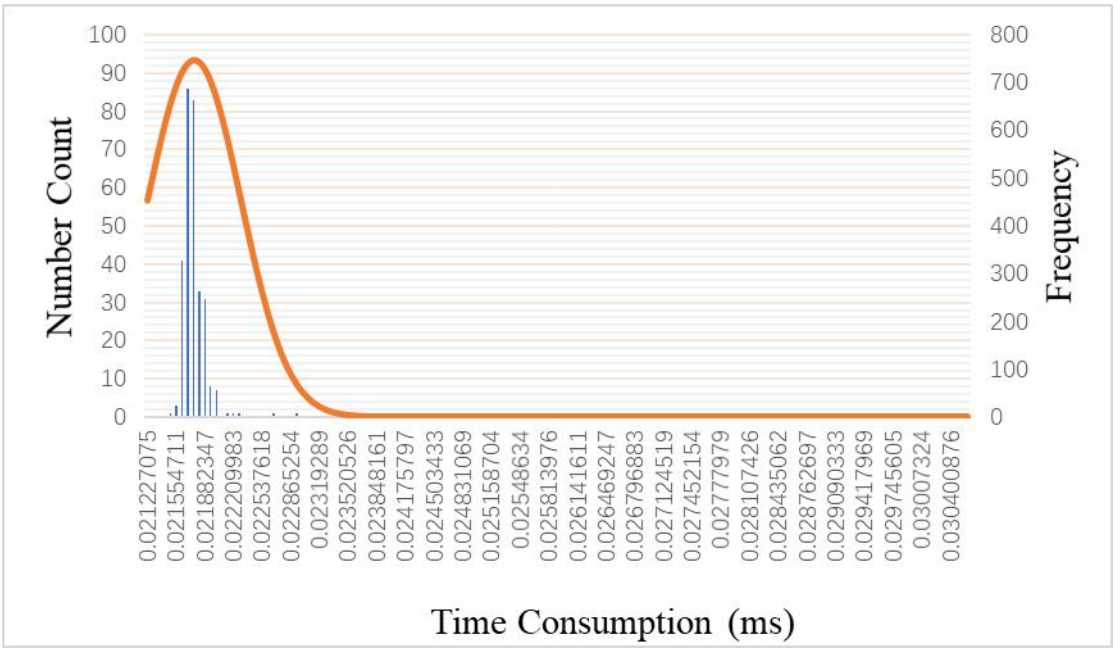

Fig 20. Time consumption of 300 data encryption each time (2 KB)

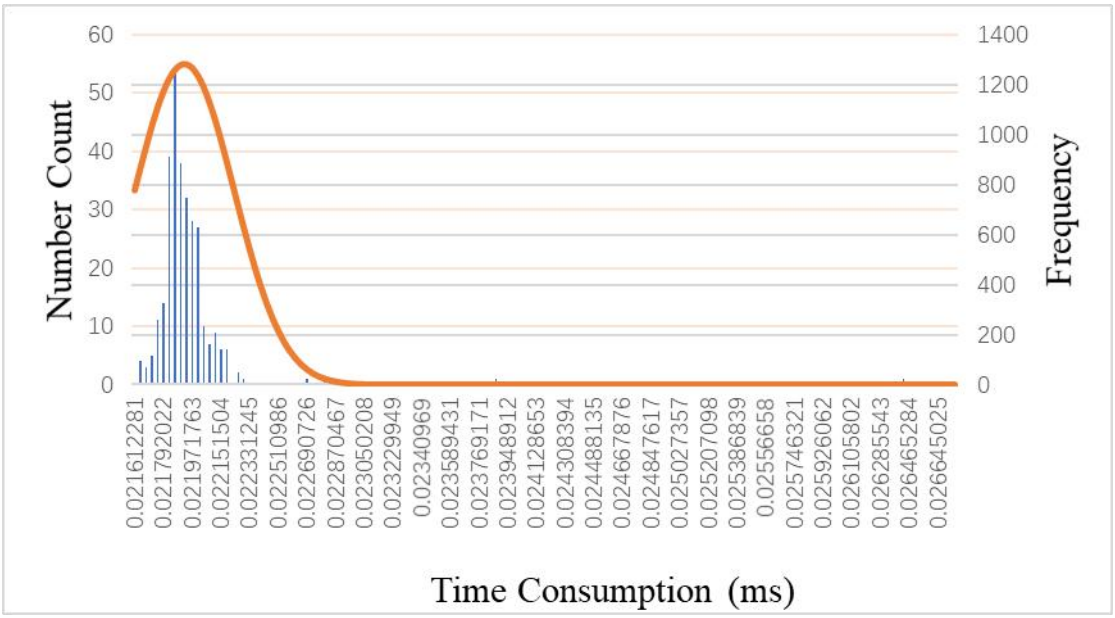

Fig 21. Time consumption of 300 data decryption each time (2 KB)

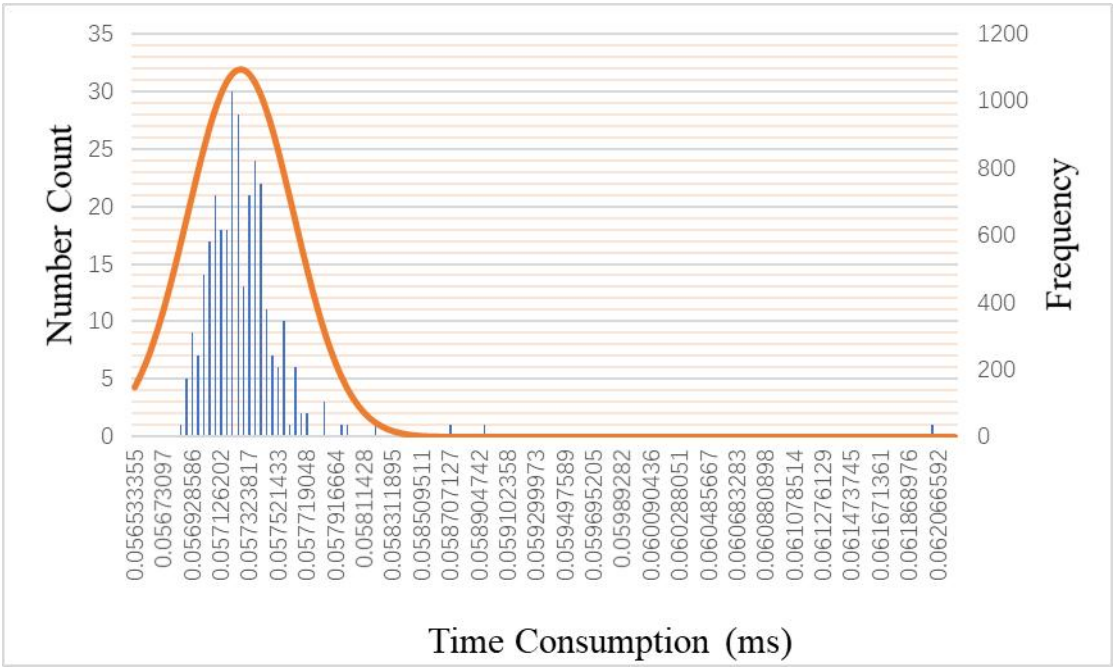

Fig 22. Time consumption of 300 data encryption each time (4 KB)

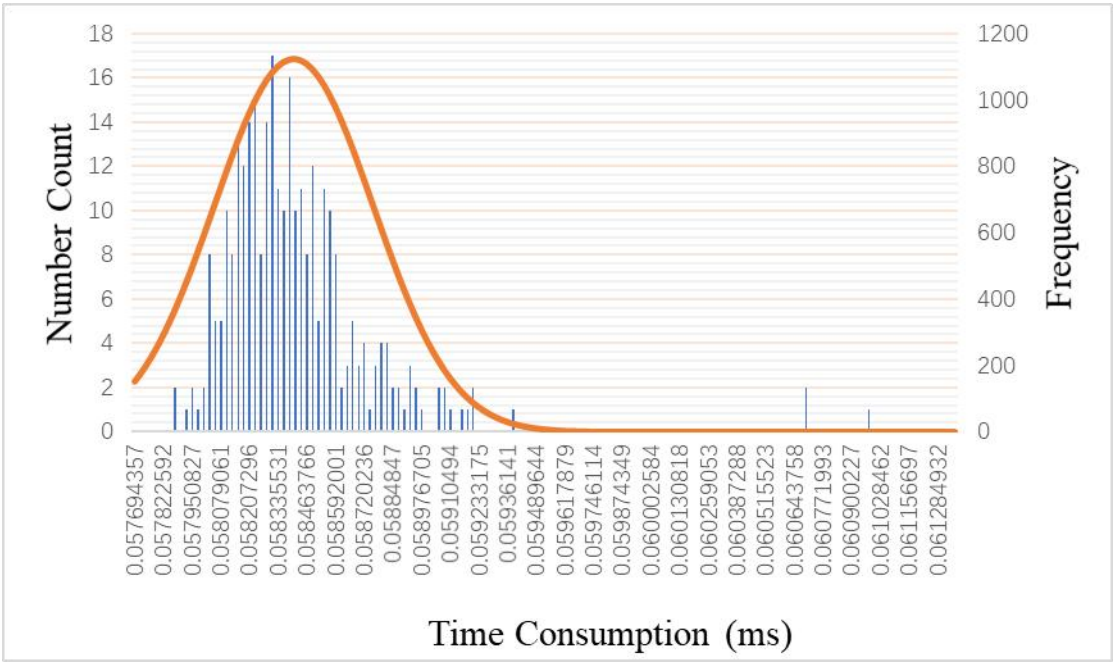

Fig 23. Time consumption of 300 data decryption each time (4 KB)

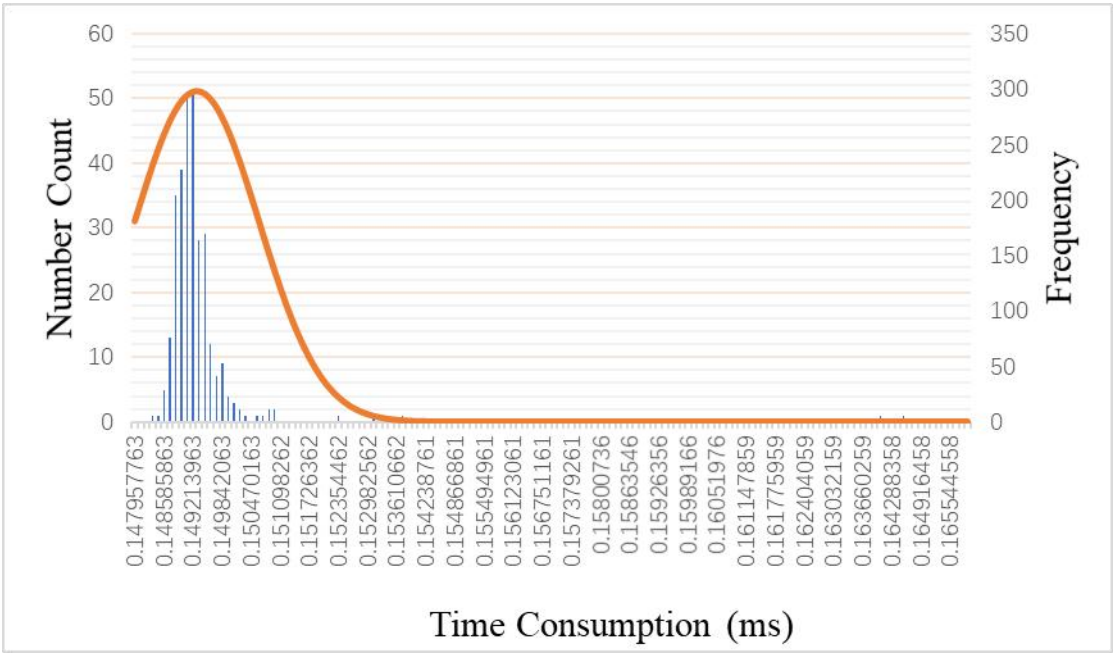

Fig 24: Time consumption of 300 data decryption each time (8 KB)

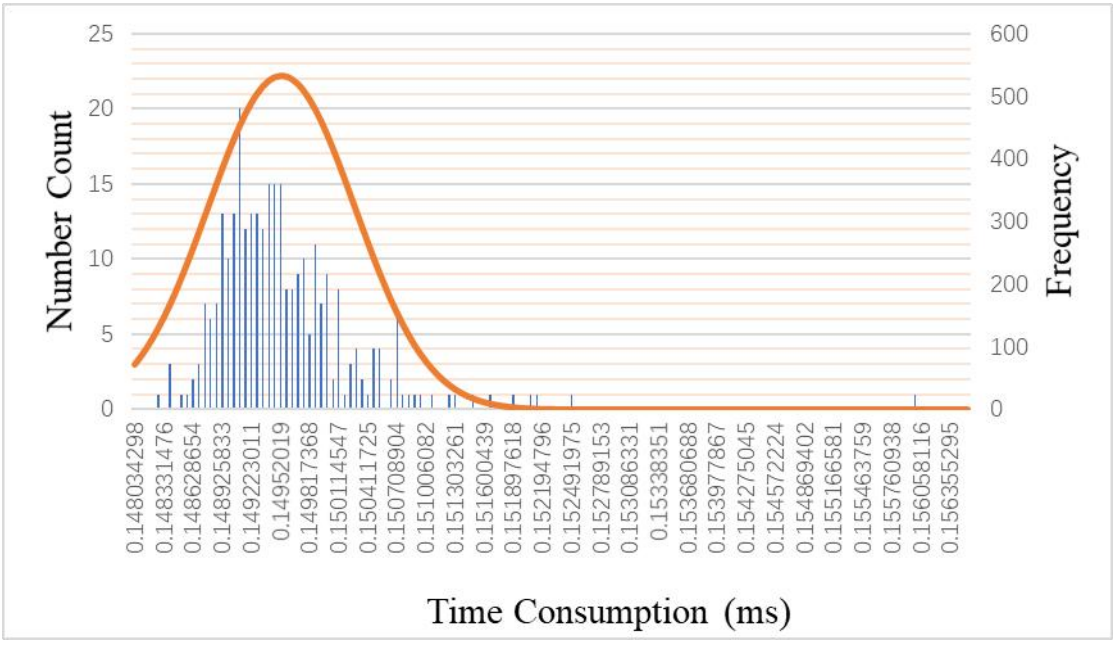

Fig 25. Time consumption of 300 data decryption each time (8 KB)

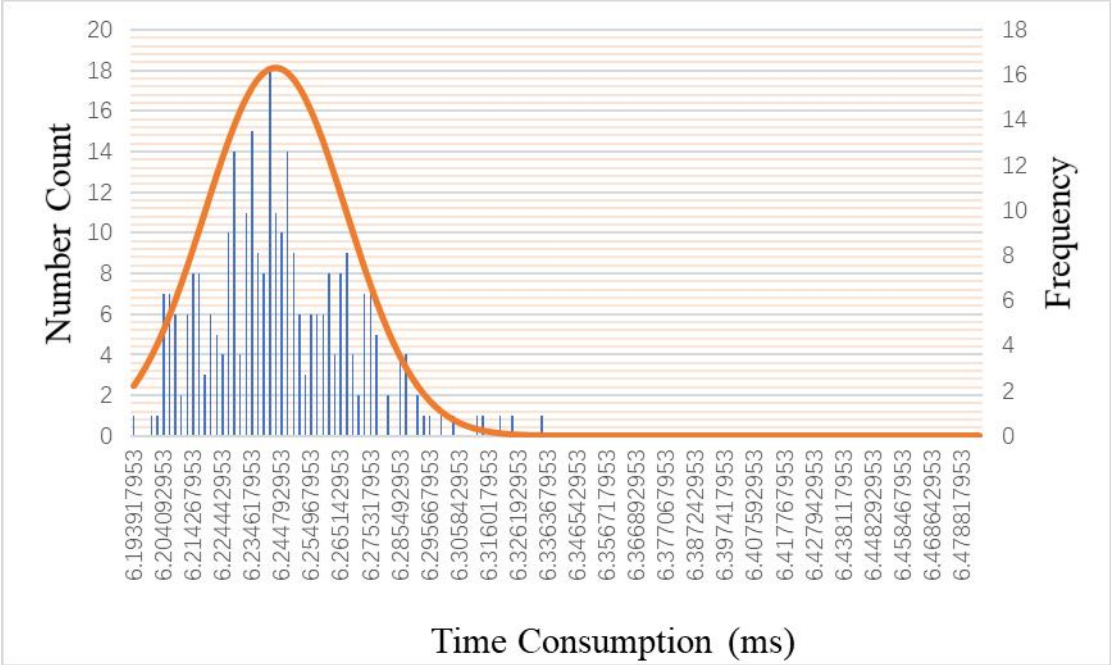

Fig 26. Time consumption of 300 data digest generation each time (800 KB)

h. Analysis of Experimental Results

From experiments a, b, and c, the PoA adopted by AuthROS has obvious advantages over PoW in multi-robots interaction scenarios. In addition, the SM4 algorithm adopted by AuthROS has excellent encryption and decryption speed and stability. The security of AuthROS will be discussed in the next chapter.

i. Safety Discussion

The security of smart AuthROS can be evaluated along the dimensions of confidentiality, integrity, and anti-attacks of data sharing.

1) Confidentiality: The Ethereum network which AuthROS used is a private Ethereum network. Thus, addresses of EOAs, which are tied with registered robots, within the network are known beforehand. Only when the EOA is known to the network and is tied with a registered robot can this EOA interact with the contract through functions within the contract. This mechanism provided confidentiality for secure data sharing.

2) Integrity: Security of SM4 key used for data encryption is guaranteed by the key exchange mechanisms. Data is encrypted by the SM4 algorithms as long as it is captured. Then the ciphertext will be encrypted by the SM4 algorithms again after a process of format conversion before transmitted to the server. After one decryption, the ciphertext will be transmitted to the Ethereum network for persistent storage. Plaintext of data is protected by two layers of encryption and robot owner can check the ciphertext stored in the Ethereum network through homomorphic encryption to ensure that is it shares, ensuring that the shared data is not tampered with.

3) Anti-attacks: The data sharing of AuthROS relies on EOA and SM4 keys in the Ethereum network. Only data that is shared in the Ethereum network is considered reliable and trustworthy. The EOA account of the robot or the robot within a swarm is fully visible to

the manager of the robot / swarm, so the manager can easily view the data captured by the robot or cluster. At the same time, the manager will only share the SM4 key with the EOA account of the trusted party so that the trusted party can view the data. Therefore, when the attacker has no EOA in AuthROS's blockchain network, the data stolen by the attacker in the local ROS will not be recognized by other members of the network. Even if the attacker has an EOA account, since the attacker is independent of the AuthROS environment and the AuthROS user doesn't know the EOA address of the stranger in the network, so, the SM4 key of AuthROS users will not be shared with the attacker. Data stolen by the attacker is only visible to themselves because AuthROS users will not recognize the data of untrusted parties.

6 Conclusion

This paper proposes an Ethereum-based data sharing framework on the base of SM algorithms for robots based on ROS. AuthROS is also equipped with a key exchange mechanism to guarantee the security of SM4 key used for data encryption and authority granting mechanism to guarantee the trustworthiness of shared data and the controllability of information data. This paper also introduces the framework of AuthROS Ethereum network, the work principles of AuthROS and the experimental platform including software and hardware. Experiments focus on the effect of consensus algorithms, message sizes, block difficulties and SM4 encryption efficiency on AuthROS.

7 Acknowledgements

This work was supported by Key Research and Development Program of Hainan Province (Grant No. ZDYF2020033), Young Talents' Science and Technology Innovation Project of Hainan Association for Science and Technology (Grant No. QCXM202007), Hainan Provincial Natural Science Foundation of China (Grant No. 2019RC107, Grant No. 2019RC098, Grant No. 621RC612), Innovation and Entrepreneurship Training Program for College Students (Grant No.202110589017X).

References

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

[2] Quigley M, Gerkey B, Smart W D. Programming Robots with ROS: a practical introduction to the Robot Operating System[M]. " O'Reilly Media, Inc.", 2015.

[3] McClean J, Stull C, Farrar C, et al. A preliminary cyber-physical security assessment of the robot operating system (ros)[C]//Unmanned systems technology xv. International Society for Optics and Photonics, 2013, 8741: 874110.

[4] Cheminod M, Durante L, Valenzano A. Review of security issues in industrial networks[J]. 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[J]. Proceedings of the IEEE, 2005, 93(6): 1152-1177.

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

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

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

[9] GM/T 0003-2012, SM2 elliptic curve public key cryptography algorithm[S].

[10] GM/T 0002-2012, SM4 block cipher algorithm[S].

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

[12] Li M, Lu K, Zhu H, et al. Robot swarm communication networks: frameworks, protocols, and applications[C]//2008 Third International Conference on Communications and Networking in China. IEEE, 2008: 162-166.

[13] Li M, Harris J, Chen M, et al. Framework and protocol design for a pervasive robot swarm communication networks[J]. Wireless Communications and Mobile Computing, 2011, 11(8): 1092-1106.

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

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

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

[17] Li J, Wu J, Li J, et al. Blockchain-based trust edge data inference of multi-robot systems for collaborative tasks[J]. IEEE Communications Magazine, 2021, 59(7): 94-100.

[18] Xia Q, Sifah E B, Smahi A, et al. BBDS: Blockchain-based data sharing for electronic medical records in cloud environments[J]. Information, 2017, 8(2): 44.

[19] Nishida Y, Kaneko K, Sharma S, et al. Suppressing chain size of blockchain-based information sharing for swarm robotic systems[C]//2018 Sixth International Symposium on Computing and Networking Workshops (CANDARW). IEEE, 2018: 524-528.

[20] Li J, Wu J, Li J, et al. Blockchain-based trust edge data inference of multi-robot systems for collaborative tasks[J]. IEEE Communications Magazine, 2021, 59(7): 94-100.

[21] Pacheco A, Strobel V, Dorigo M. A Blockchain-Controlled Physical Robot Swarm Communicating via an Ad-Hoc Network[C]//International Conference on Swarm Intelligence. Springer, Cham, 2020: 3-15.

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

[23] Zhaohui Wang, Zhenfeng Zhang. Summary of SM2 Elliptic Curve Public Key Cryptography Algorithm[J]. Research on Information Security, 2016, 2(11): 972-982.

[24] Shuwang Lv, Bozhan Su, Peng Wang, et al. Summary of SM4 Block Cipher Algorithm[J]. Research on Information Security, 2016, 2(11): 995-1007.

[25] Caiazza G. Application-level security for robotic networks[J]. 2021.

[26] Kircanski A, Shen Y, Wang G, et al. Boomerang and slide-rotational analysis of the SM3 hash function[C]//International Conference on Selected Areas in Cryptography. Springer, Berlin, Heidelberg, 2012: 304-320.