Computers, Materials & Continua CMC, vol.65, no.1, pp.723-741, 2020

A Distributed Privacy Preservation Approach for Big Data in Public Health Emergencies Using Smart Contract and SGX

Jun Li[1, 2], Jieren Cheng[2, *], Naixue Xiong[3], Lougao Zhan[4] and Yuan Zhang[1]

Abstract: Security and privacy issues have become a rapidly growing problem with the fast development of big data in public health. However, big data faces many ongoing serious challenges in the process of collection, storage, and use. Among them, data security and privacy problems have attracted extensive interest. In an effort to overcome this challenge, this article aims to present a distributed privacy preservation approach based on smart contracts and Intel Software Guard Extensions (SGX). First of all, we define SGX as a trusted edge computing node, design data access module, data protection module, and data integrity check module, to achieve hardware-enhanced data privacy protection. Then, we design a smart contract framework to realize distributed data access control management in a big data environment. The crucial role of the smart contract was revealed by designing multiple access control contracts, register contracts, and history contracts. Access control contracts provide access control methods for different users and enable static access verification and dynamic access verification by checking the user's properties and history behavior. Register contract contains user property information, edge computing node information, the access control and history smart contract information, and provides functions such as registration, update, and deletion. History contract records the historical behavior information of malicious users, receives the report information of malicious requestors from the access control contract, implements a misbehavior check method to determines whether the requestor has misbehavior, and returns the corresponding result. Finally, we design decentralized system architecture, prove the security properties, and analysis to verify the feasibility of the system. Results demonstrate that our method can effectively improve the timeliness of data, reduce network latency, and ensure the security, reliability, and traceability of data.

Keywords: SGX, big data, privacy protection, smart contract, access control.

1 Introduction

In the age of information explosion, the value of data has undoubtedly contributed to the

[1] Hainan Blockchain Technology Engineering Research Center, Hainan University, Haikou, 570228, China.

[2] School of Compute Science and Cyberspace Security, Hainan University, Haikou, 570228, China.

[3] Department of Mathematics and Computer Science, Northeastern State University, Tahlequah, 74464, USA.

[4] Key Laboratory of Internet Information Retrieval of Hainan Province, Hainan University, Haikou, 570228, China.

[*] Corresponding Author: Jieren Cheng. Email: cjr22@163.com.

Received: 29 April 2020; Accepted: 29 May 2020.

development of big data technology. The value of data is essential for a wide range of scientific and industrial processes. The digitization of workflows is the current big trend in the medical industry. Then, as this trend shifts, the amount of data that was obtained electronically will increase dramatically, and these electronic medical data are complexity, diversity, and mass. Medical data is an important part of big data and plays a critical role in public health emergencies, for example, in recent days, coronavirus disease (COVID-19) has become a global epidemic, and the doctor needs real patient data to make a rapid clinical decision. Nevertheless, different countries had devastating data loss because of the large cyberattack on medical systems [Lawless (2017)]. A key aspect of public health emergencies is to guarantee medical organizations must effectively protect patient privacy and data security. The security of medical data is essential for big data in public health emergencies, which will improve data privacy and rapid clinical decision-making. Jiang et al. [Jiang, Coffee, Bari et al. (2020)] studied the effects of building an artificial intelligence framework to algorithmically identify the combinations of clinical features of COVID-19 on providing rapid clinical decision-making support. However, previous research has established that due to the complexity, diversity, and quantity of data, use traditional software or hardware to manage these data is difficult, especially the security and privacy of data [Chen, Yang, Hao et al. (2017); Khalil (2012)]. A considerable amount of literature has been published on data privacy protection. These studies put forward the location data record privacy scheme to protect highly frequent accessing location data [Gu, Yang and Yin (2018)], local privacy protection method [Yin, Zhou, Yin et al. (2019); Yin, Ju, Yin et al. (2019)], and an efficient collaborative filtering algorithm for privacy protection based on differential privacy protection and time factor [Yin, Shi, Sun et al. (2019)]. He et al. [He, Zeng, Xie et al. (2017)] proposed a distributed privacy protection scheme in the smart grid environment, which used the homomorphic encryption function to reduce the complexity of forwarding nodes and solved the privacy protection problem of SG in random linear network coding. Nevertheless, different from these studies, this paper utilizes smart contracts to provide a distributed hardware-enhanced privacy protection method. Nowadays, huge electronic medical big data is stored in the medical cloud. But there are different security issues in health cloud computing. Recent evidence suggests that the most important of security issues are: legal and policy issues, data protection, privacy protection, lack of transparency, network security issues, lack of security standards, software licensing [Sargita, Ankita and Reshamlal (2015)]. Among them, data security and privacy protection are an urgent problem to be solved.

On the other side, researchers have shown an increased interest in moving from the central cloud to the edge of the network [Satyanarayanan (2017)]. Edge Computing [Shi and Dustdar (2016)], as an extension of the cloud, enables billions of different devices to perform data computing, storage, and running applications at the edge of the network [Shi, Cao, Zhang et al. (2016)]. The distributed structure of edge computing has many benefits. Edge computing helps users by providing computing power, data storage, and application services, reducing transmission latency during data computing. However, surveys [Yu, Liang, He et al. (2018)] have shown that the security and privacy of edge computing nodes that operate on different, untrusted third-party devices remain a huge challenge. Traditional data is stored on a local computer or cloud server, unfortunately, this approach results in problems related to attack. For example, the number of distributed denial-of-service (DDoS) attacks on mainstream cloud

platforms is increasing. Cheng et al. [Cheng, Li, Tang et al. (2018)] proposed a genetic algorithm based on flow correlation (FCD) characteristics to optimize DDoS attack detection methods that enhance random forests (RF). This method can effectively detect DDoS attacks in the cloud environment, with higher accuracy and lower false-positive rate. Previous research has found a more secure and generic option to improve the security of data storage is to use a trusted execution environment (TEE) [Zhang, Cecchetti, Croman et al. (2016)]. A common trusted execution environment is SGX. SGX aims to enable high-level protection of secret data and code. From the perspective of SGX technology, SGX provides a completely isolated environment called enclave that prevents other applications, operating systems, and host owners from tampering with critical data and code. To protect data during execution, SGX provides a security mechanism that allows each enclave to encrypt and authenticate data for persistent storage. Through security mechanisms, SGX can develop a variety of security-enhancing applications for its privacy-enhancing analytics, hardware-enhanced data protection, and assist with the edge computing. But there is still a critical flaw that SGX was unable to effectively manage users' data access rights to a wide range of resources. To solve this problem, the smart contract can be used for secure data access management solutions. Smart Contract is an automated, programmable computer protocol designed to help you exchange money, property, or anything of value in a transparent, conflict-free, and trusted manner without trusting a third party. Different methods have been proposed to solve access control issues for resources. Traditional access control approaches including the capability-based access control (CapBAC) [Hernández-Ramos, Jara, Marín et al. (2013)], the role-based access control (RBAC) [Kuhn, Coyne and Weil (2010)], and attribute-based access control (ABAC) [Hu, Kuhn, Ferraiolo et al. (2015)]. However, most of these traditional solutions rely on centralized physical institutions that cannot provide an efficient mechanism for easy management of the spontaneity, scalability, and heterogeneity of the system. Hence, the smart contract has emerged as an interesting candidate to control network behavior and prevent malicious users because of its impressive distributed and trustworthiness.

Data privacy means that the data will be processed privately or authorization is needed to access the data. Data security is the data that is impervious, reliable, and traceable. For decades, however, protecting that data has been a daunting task for medical organizations, especially in public health emergencies, not only to ensure patient privacy and data security but also to calculate, store, and transmit data securely. For this study, it was of importance to investigate whether the combination of SGX and smart contracts can provide strong security and privacy of big data in public health emergencies. Therefore, this paper proposed a distributed privacy preservation approach combined with SGX and smart contracts. Based on SGX, we design data protection modules using symmetric encryption which has high speed and efficiency. We define access control policies based on smart contracts to achieve access control management for big data in public health emergencies. The experiment shows that this method has high timeliness, low latency, and achieves reliable, traceable and secure storage of data.

2 Background

In this section, we will briefly introduce background on the main technologies of the system, namely SGX, edge computing, and smart contract.

2.1 SGX

SGX is a set of CPU instructions that enable user applications to create protected areas in the app address space, called enclaves [Intel Corporation (2015)]. The enclave is isolated from untrustworthy system software, allows the secure operation of legitimate applications to be encapsulated in an enclave [Sawtooth (2016)], and ensures a malicious OS cannot read or modify enclave memory at runtime. SGX adds memory access mechanisms to provide a high level of protection for confidential information, trusted hardware for users, and protection against malware attacks. Only the CPU or the application itself can get the right to access the code and data in the enclave, software outside the enclave, including privileged software (such as VMM, BIOS, and OS) and nonprivileged software cannot access data inside the enclave. The enclaves use ecall/ocall interfaces to switch control between the trusted part of the app and the untrusted part of the app [Intel Corporation (2016)]. Also, data that needs to be re-used later will be protected by using Intel protected file system library. SGX provides remote attestation which allows a remote host to check if the application running in the enclave is legitimate. Attestation aims to check whether the software in an enclave has been properly instantiated on the platform [Anati, Gueron, Johnson et al. (2013)]. Once the attestation is successful, the secure channel will be made between the enclave. When the enclave process exits, the enclave will be destroyed and any data that is secured within the enclave will be lost. To protect enclave data across executions, SGX provides the application programming interface (API) that allows each enclave to encrypt and authenticate data for persistent storage outside the enclave, such as on disk. Through isolation, sealing, attestation of these security mechanisms, SGX was able to help increase privacy and security for data processing, enable isolation computation on sensitive data, and develop various services with hardened security. For more information, please see [Costan and Devadas (2016)], which provides a thorough introduction to the SGX architecture and security analysis of SGX.

2.2 Edge computing

Edge computing is the technologies that deploy edge nodes between the cloud server and the end users, allow the computation to be performed at the network edge. The network edge server can support most of the traffic on the network as well as a large number of resource requirements, such as real-time data processing and computation offload. As a result, network edge servers provide better performance for end users, while small latency increases. Recently, there has been an increasing amount of literature on edge computing. Edge computing has been used in many scenarios. For example, research on edge computing has been carried out in industrial robotics systems [Chen, Feng and Shi (2018)], the author's experiments have demonstrated that the system has better real-time and network transmission performance than cloud-based scenarios. Video analysis [Yi, Hao, Zhang et al. (2017)], the author proposed LAVEA, an edge computing system that can offload computing tasks between clients and edge nodes, providing users with low latency. One study by Tang et al. [Tang, Wang, Song et al. (2019)] examined the trend toward charging stations for electric vehicles, propose the charging and discharging networking system algorithm to minimize the waiting time for electric vehicles. A recent study by Li et al. [Li, Chen, Gao et al. (2018)] provides the multi-model framework based on RSS to

solve indoor localization problems under the mobile edge computing environment. Smart firefighting [Wu, Dunne, Zhang et al. (2017)], the author uses the edge calculation low delay this important feature, set up fire simulation system, the experimental results demonstrate that the system reduces the system delay by 50%. As a result, edge computing shifts a large amount of traditional cloud computing resources and services to edge nodes, reduces access latency to improve the user experience, and addresses resource constraints for resource-constrained devices. Edge computing is a key technology to realize the vision of the next generation internet, such as the tactile internet [Aijaz, Dohler, Aghvami et al. (2017)], which review some of the rigorous design challenges and suggest the first way to achieve a specific solution to the tactile internet revolution.

2.3 Smart contract

A smart contract is a self-executing, programmable computer protocol intended to help you exchange money, property, or anything of value in a transparent, conflict-free way. Bhargavan et al. [Bhargavan, Delignat-Lavaud, Fournet et al. (2016)] introduced the concept of smart contracts. They allow parties that do not trust each other to execute agreements expressed in the Solidity programming language without involving any third parties. In the blockchain environment, the smart contract is a script stored on the blockchain. Users can interact with smart contracts by predefined public functions or application binary interfaces (ABIs). The purpose of smart contracts is to provide a secure method that is superior to traditional contracts and to reduce other transaction costs associated with contracts. The code in a smart contract contains a set of rules under which the parties to the smart contract agree to interact with each other. If a predefined rule is met, the protocol will be performed automatically. Smart contracts provide a mechanism to effectively manage access between assets and two or more participants. Values and access rights are stored on the blockchain, which is a transparent shared ledger that protects them from deletions, tampering, and revisions. Smart contracts have multiple attractive usage scenarios, such as financial contract [Biryukov, Khovratovich and Tikhomirov (2017)], elections [McCorry, Shahandashti and Hao (2017)], auctions [Hahn, Singh, Liu et al. (2017)], access controls [Azaria, Ekblaw, Vieira et al. (2016)], and trading platforms [Notheisen, Gödde and Weinhardt (2017); Mathieu and Mathee (2017)]. As a result, smart contracts provide an open and verifiable way to embed governance rules and business logic in code that can be reviewed and executed by majority consensus on P2P networks. The smart contract framework workflow is based on the Ethereum platform, for a detailed introduction to the Ethereum platform, please refer to [Ethereum community (2016)], which provides history, community, and guidelines.

3 System architecture and security model

In this section, we presented the system architecture and security model.

3.1 System architecture

An architectural schematic of the system showing its interaction with external entities is given in Fig. 1. As illustrated in Fig. 1, the system considered in this paper consists of a large number of different users, storage devices, and user devices, which are connected through the network. Also, present in the system are numerous edge nodes, which are

connected to the resource-constrained devices and blockchain network via the secure channels. The basic structure of this architecture is logically simple: the users use resource-constrained devices to establish a trusted connection with SGX, and then the user sends request to SGX. When SGX receives the request, it will send the request to smart contracts to check whether the user is allowed to access the data. Finally, the access control results will be returned to SGX, then SGX sends the data to users based on access control results. The system architecture consists of three parts, each part is explained as follows.

Users: There are multiple types of users, which can be patients, doctors, nurses, and other users. These users can use different devices, such as mobile phones and computers, to make requests to and enjoy the services provided by the edge computing nodes.

Edge computing nodes: Here SGX act as an edge computing node which has three main functions: Firstly, it interacts with the user devices to provide a variety of services for users. Secondly, it acts as a trusted data input for smart contracts, ensuring that data is not compromised. Thirdly, it provides privacy-enhancing data encryption and stores patients' private data safely.

Smart contract: The smart contract, which is deployed on Ethereum's blockchain platform, mainly includes multiple access control contracts (ACC), one history contract (HC), and one register contract (RC).

3.2 Security model

Here a brief security model for the system is described.

- Trusted nodes: Trusted nodes are nodes that haven't been attacked. But once the trusted nodes are attacked, it will have the possibility to steal the source data and leak the user's privacy.
- SGX: It should be pointed out that we make two assumptions: (1) The enclave is trusted and the source code and data are executed correctly. (2) The hardware is not compromised by a malicious attacker.
- DDoS attack: Attackers may control many resource-constrained devices to invoke the SGX repeatedly in the system, thus prevents SGX to serve non-attackers.
- Network communication: The attacker controlling the network may tamper with or delay the transmission during the session between the resource-constrained devices, SGX, and blockchain nodes.
- Ethereum: The integrity and confidentiality of the transactions are guaranteed, but not sure.

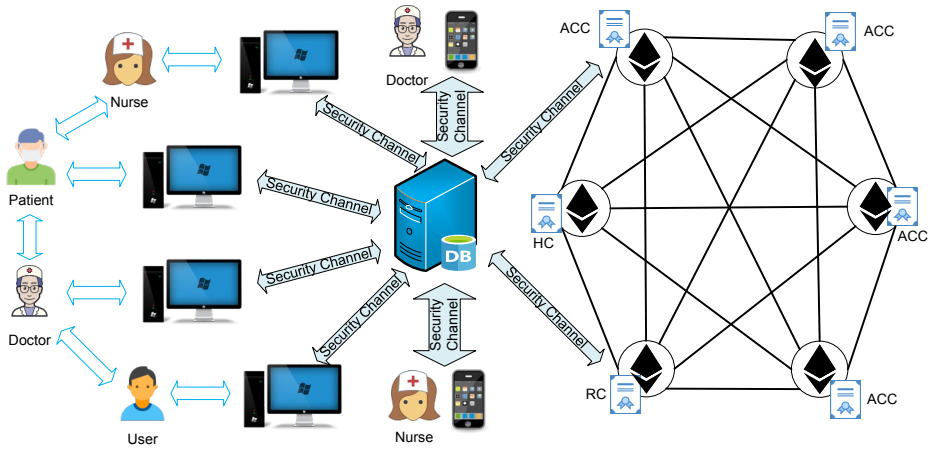

Figure 1: System architecture

4 System design

This system is implemented on the real Ethereum blockchain test network. Compared to other open blockchain platforms such as Bitcoin and Hyperledger, Ethereum is an open source project that is highly adaptable and flexible, built by many people, and is designed to enable developers to create arbitrary consensus-based, scalable, standardized, feature-complete, easy-to-develop, and collaborative applications [Buterin (2014)].

4.1 Smart contract framework and workflow

The smart contract framework is illustrated in Fig. 2. In the following, we carefully explain the role of the framework. The proposed framework is based on the Ethereum platform and consists of multiple ACCs, one HC, and one RC. Moreover, smart contracts are securely stored in the blockchain, and only users can change their contracts. The introduction of each contract is as follows.

ACC: An ACC is deployed by a creator who wants to execute an access control management of data. Here we assume that the user has access control permissions that can negotiate multiple data, and each access policy is used by an ACC. Therefore, a user can be associated with multiple ACC. In this framework, to control requests from the user, each ACC can not only use static transaction access authorization verification by checking predefined policies but also enable dynamic verification of transactions by checking the real-time behavior of the user. ACC is composed of the following key components:

- Resource: The policy defines the data resource in public health emergencies, such as the data file.
- Action: The action that is performed on the resource, such as read, write, etc.
- Permission: The static permission, such as allow, deny, etc.
- Timestamp: Identifies the last time the requester accessed. At the same time, the timestamp is used for dynamic validation, such as checking if the requester sent an access request frequently in a short period of time.

- ABI: The ACC provides the ABI to manage and enforce access control policies, mainly including policyAdd, policyDelete, policyUpdate, accessControl, deleteACC, etc. Note that only the creator of the ACC can manage these ABIs.

HC: To check whether the user is malicious requestors during the data access control process, the check method of history misbehavior was used. HC checks the corresponding result when it receives the report of potential malicious requestors from the ACC. The check result is based on the user's access history of misbehavior, so HC may need to record historical access information of all malicious users. After determining the check result, HC returns the decision to the ACC for further action. HC includes the following key parts:

- Misbehavior: It includes the record of misbehavior history, misbehavior check, etc.
- Time: Time was used as an aid to record the time when a malicious requester accessed.
- ABI: ABI using misBehaviorRecord, misBehaviorCheck, deleteHC, etc. Any ACC can run the misBehaviorCheck ABI to HC to check requestor history misbehavior. The check result will be based on the history of the misbehavior of the requestor, and then returns the result to ACC who reported the misbehavior. This ABI also adds a new misbehavior record to the misBehaviorRecord.

RC: The main role of the RC in the system is to manage registration information for user properties and edge nodes, access control contracts, and history contracts. The detailed of RC is listed as follows:

- MethodName: The name of each method.
- ScName: The name of the corresponding smart contract that implements this method.
- ScAddress: The address of the smart contract.
- ABI: The ABIs include methodRegister, methodUpdate, methodDelete, getContract, etc.
- Creator: The peer who created and deployed the contract. Note that only the method creators can register, update, and delete the method.

In the smart contract framework, the details of the access control workflow are as follows:

- Step 1: The user uses the device to send a data request to SGX.
- Step 2: When SGX receives a user request, it firstly calls the getcontract ABI of RC from blockchain to retrieve the address and ABIs of the deployed access control contracts.
- Step 3: The RC returns the address and the ABI of the ACC to the SGX, and the SGX analyzes the specified data received.
- Step 4: SGX transfers the data which contains the information needed for access control to call the ACC's access control ABI into the smart contract through a trusted security channel.
- Step 5: In the process of dealing with the access control policy, if some potential misbehavior is detected, the ACC will send a message to HC, calling HC's misbehaviorcheck ABI.
- Step 6: Once the misbehavior ABI checks the access history for misconduct and determines the result, HC will return the result to the ACC.
- Step 7: Finally, the access control results will be returned to SGX.
- Step 8: To determine the access control process results, if the access right policies are

met, the SGX grants the access request and offers services to the user. Otherwise, the service request is denied.

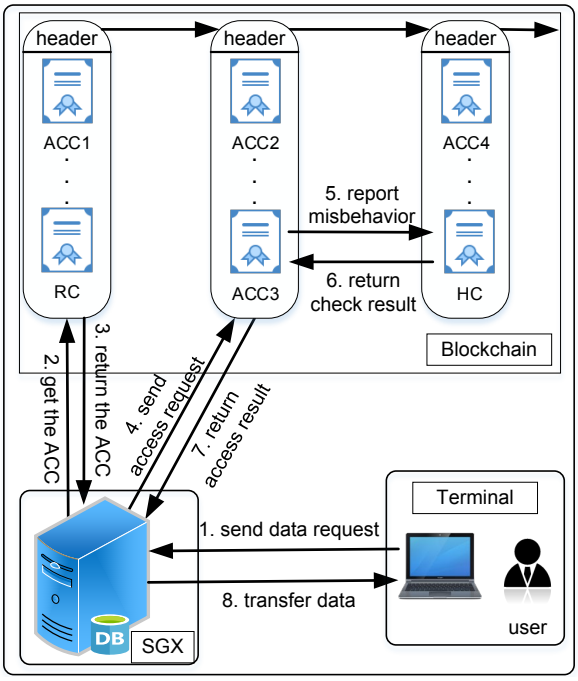

Figure 2: Smart contract framework

Access control is a big security concern in data protection. The ability to manage and leverage data access control is essential to the success of all medical industries. Because the access control policy is implemented through the form of smart contracts and stored in the blockchain, the policy information is verifiable, traceable, and cannot be maliciously tampered to anyone, thus effectively improving the credibility of the system. Note that only the creator of ACC can add new policies, update or delete existing policies, and manage HC as well as ACC.

4.2 TEE design

The overall design structure of the SGX platform is described in this part. In the SGX trusted platform, untrusted modules and trusted modules interact through the interfaces of ecall/ocall. In the enclave, we designed three sub-modules, which are the data access module, data protection module, and data integrity check module. The privacy protection code is written in the enclave, formulates data access policies to assure that data privacy is complete, autonomous, and controllable. After obtaining the specified access permission, the private data will be decrypted in the enclave and complete the corresponding operation after performing the data integrity check. Because the enclave lacks networking capabilities, all communication between the resource-constrained devices and the enclaves is done using transport layer security (TLS). A major advantage of TLS is the fact that it can encrypt the transmitted data to ensure the integrity and security of the data.

Data access module: The purpose of this module is to check whether ACC grants the requester data access. When the user requests data, it needs to connect with SGX, provide the user message. SGX then establishes communication with the smart contract, providing user messages, and waiting for the result from ACC. If the current result is validated successfully, proceed to the next step. Otherwise, if the current result validation fails, the data validation process will stop, and the access request will be denied. This module will follow the predefined access rules to ensure that the correct action is carried out.

Data protection module: Several techniques have been developed to protect the sensitive data in the enclave. The Intel protected file system library using software guard extensions is the main non-invasive method used to protect the important file. There are many file operation function APIs provided by Intel SGX available for creating, operate, and delete files inside the enclave. The benefit of using the API is that files are encrypted during write operations and stored on an untrusted disk, and their confidentiality and integrity are verified during reading operations. So, based on Intel protected file system library, we design the file protection method using symmetric encryption to keep the confidential data secure and safe in the SGX trusted platform.

Data integrity check module: Even if the requester gains access to the target data, it is necessary to perform the data integrity check method through the data integrity check module to prevent the data from being tampered with by a malicious attacker. The security steps performed by the data integrity check include generating a password s, calculating its hash value $h = Hash(s)$ for the private data. If the hash value does not change, we assume that the data has not been attacked.

5 Evaluation analysis

In this section, we will introduce the experiment. The standardized devices use an Intel Core i7-9700 K, 3.60 GHz, 16 GB memory machine, Windows 10 Home (64 bit), Visual Studio 2017 Pro, and Ubuntu 16.04 LTS with Linux kernel 4.4. Moreover, the environment where the Intel SGX SDK v2.4.100.51291 for Windows is installed in the operating system. Note that experimental results are based on the data from global coronavirus data [Coronavirus (COVID-19) data hub (2019)], identified features on case type, date, country, etc.

5.1 SGX response time

Tab. 1 shows an overview of the response time of SGX. SGX response time refers to the time when a resource-constrained device makes a request to SGX, and SGX responds when it receives the request. Data were gathered from four separate experiments, each of which was tested 25 times. The table below illustrates the results of the four experiments, among them, response time equals to the mean plus the standard deviation. All times in this table are in milliseconds. From the Tab. 1, it can be seen that the average response time of SGX was significantly stable at 4 ms, which is much less than 0.1 s. For small and medium-sized sites, response times below 0.1 seconds give users a sense of instant response [Nielsen (2012)]. This result may be explained by the fact that we designed a TLS transmission system based on SGX in the local environment. It is apparent from this table that high delays are rare. From these data, we can see that SGX has very short response times and low latency.

Table 1: SGX response time

Time	t_{min}	t_{max}	Response time
1-25	3.872	4.293	4.055±0.126
26-50	3.880	4.296	4.021±0.123
51-75	3.870	4.280	4.069±0.128
76-100	3.875	4.321	4.067±0.116

5.2 Encryption and decryption workload

In the course of this experiment, different file sizes played an important role. Fig. 3 compares the encryption and decryption workloads between different file sizes in the enclave. When SGX encrypts and decrypts files, it consumes CPU resources, and here we use CPU time as a measure of workload. The red line represents the encryption time and the blue line represents the decryption time. The experiment compares the workload of encryption and decryption between CSV and JSON files. From Fig. 3, we can see that as the file size grew, SGX took longer to encrypt and decrypt the file. Encryption workload from the COVID-19CSV file can be compared with the data in the COVID-19JSON file, which shows no significant difference between the two groups. Here we take the COVID-19CSV file as an example to illustrate. When the file size is 10 MB, encryption time is 28 ms and decryption time is 7 ms. When the file size is 40 MB, the time is 115 ms and decryption time is 14 ms. When the file size is 130 MB, the time is 390 ms and decryption time is 20 ms. The result shows that overall encryption and decryption time is slowly growing as the file grows, however, it will not affect the overall performance of the system. Symmetric encryption used in enclaves is fast, secure, and low in resource consumption. This finding, while preliminary, suggests that the encryption and decryption workload is very low when encrypting and decrypting files on the SGX platform.

5.3 Communication overhead

Fig. 4 provides the experimental data on communication overhead between the SGX and resource-constrained devices. This experiment also compares the different communication overhead between different format files, we tested the files in CSV format and JSON format, each format has four files of different sizes. As shown in Fig. 4, overall, the communication overhead of the COVID-19CSV file is a little higher than that of the COVID-19JSON file. The file sizes we tested ranged from 10 MB to 130 MB, and the time overhead required for both types of files fluctuated in the range of 0.1 seconds to 1.4 seconds. Taking the COVID-19CSV file as an example, when the size of the data is 10 MB, the time is close to 0.12 seconds. When the size of the data is 80 MB, it takes 0.92 seconds. When the data is 130 MB, and the time is 1.4 seconds. Compared to traditional TLS transports, the additional time in communication overhead is mainly spent on enclave creation and file encryption and decryption. Besides, SGX as an edge computing node makes the system have a large number of computing and storage resources distributed at the edge of the network, thus effectively reducing the storage and computing burden of resource-constrained devices. It is notable that the communication overhead of the experiment is built on the local platform, not on the remote platform.

5.4 Throughput performance

The purpose of this experiment was to test how many users request the enclave can handle. The experimental results are shown in Fig. 5. The experiment is divided into two groups, one group reusing the enclave, and the other one not reusing the enclave. From Fig. 5, we can see that in all two cases, the reuse group handled significantly more requests than without enclave reuse groups. Meanwhile, the user request waits for less than 5 ms. When the enclave is reused, the throughput per second is 200. In contrast, when the enclave is not reused, the throughput per second is only 25. And user requests wait time fluctuates but closer to 40 ms. This relationship may partly be explained by whether to recreate enclaves. Recreating the enclave takes a certain amount of time. In the current experiment, comparing the reuse enclave with not reuse enclave at least hint that the difference in throughput performance and stability depends on whether to reuse the enclave. It can therefore be assumed that reuse the enclave is a good way to improve throughput performance and stability of the enclave.

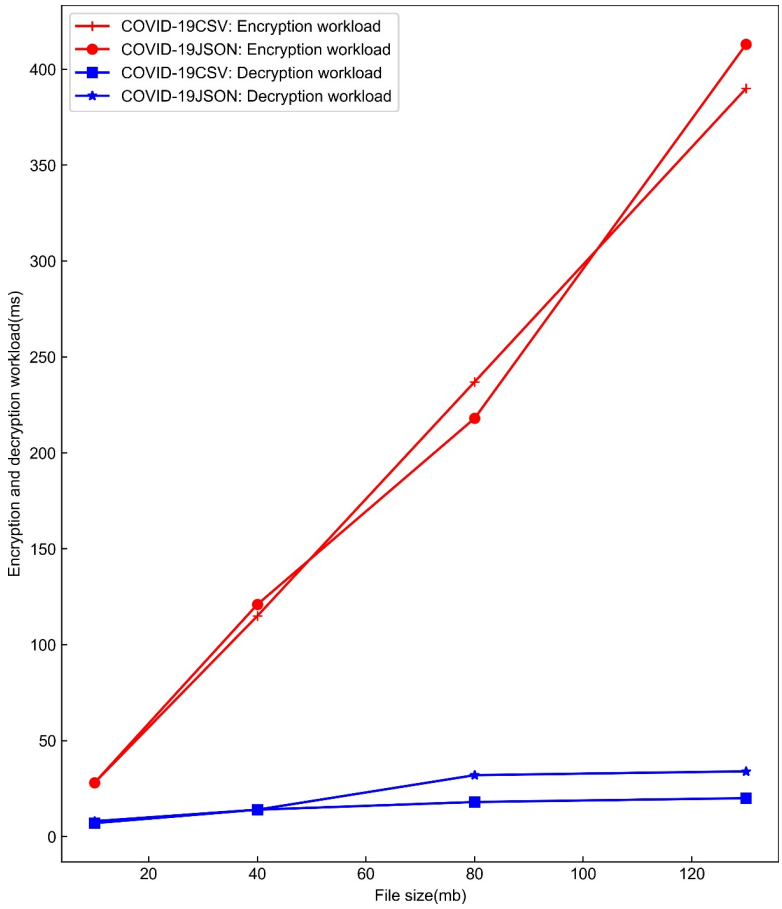

Figure 3: Encryption and decryption workload

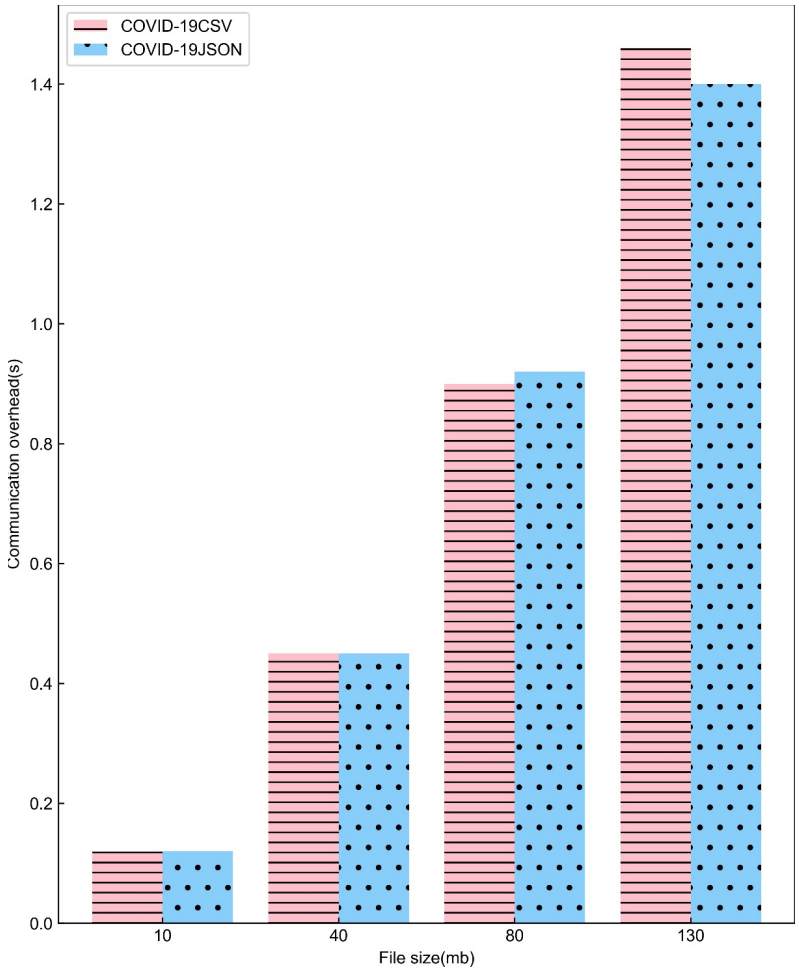

Figure 4: Communication overhead

5.5 Gas cost

To prevent the system from getting out of control due to malicious programs, executing all programs in Ethereum requires a unit called Gas as a payment to perform specified tasks. Various operating costs are calculated in gas units. Any block can calculate the amount of gas consumed according to the rules.

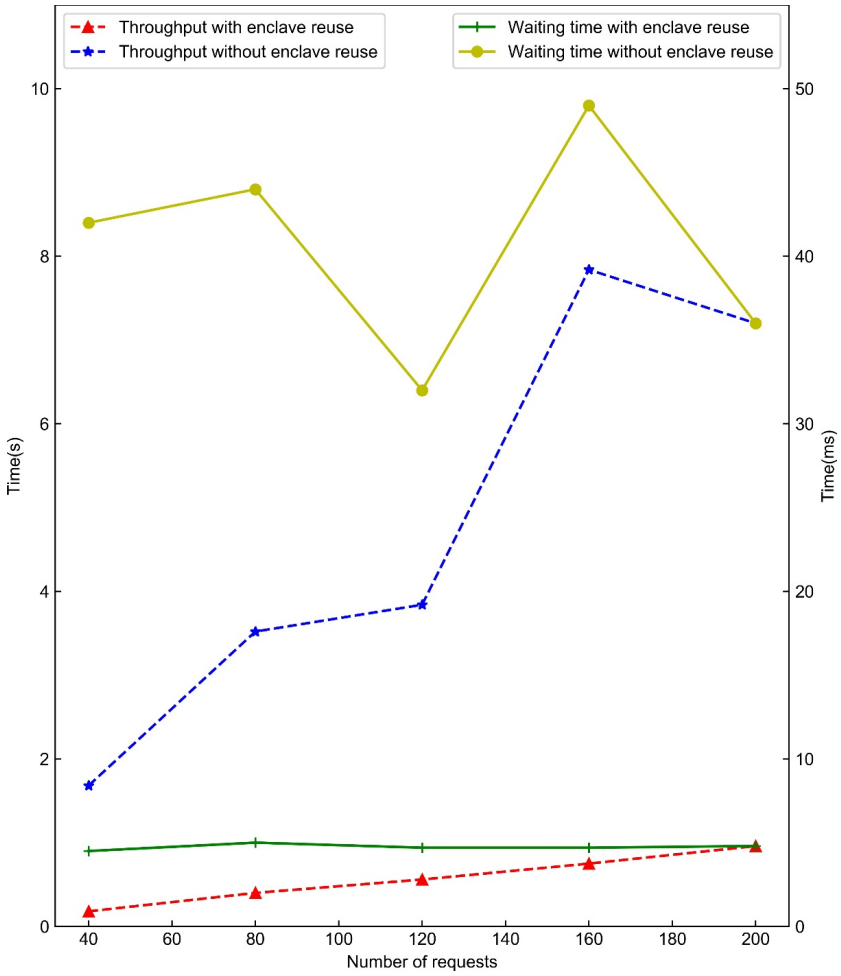

Figure 5: Throughput performance

In general, the more complex the task, the more gas is required. In this experiment, we deployed smart contracts in two environments, the official Ethereum testnet Ropsten and the test chain built locally. As shown in Fig. 6, the number of gas required to deploy RC, ACC, and HC on Ropsten is 1,326,763, 2,100,671, and 1,110,638, respectively. The number of gas required to deploy RC, ACC, and HC on the local test chain is 1,620,511, 2,561,607, and 1,365,698, respectively. A comparison of the two environmental results reveals the different gas consumed. A possible explanation for this might be that the gas value in the experimental results is affected by parameters such as gas prices and limits. Therefore, the results of experiments with different parameter settings may differ from the results of this experiment.

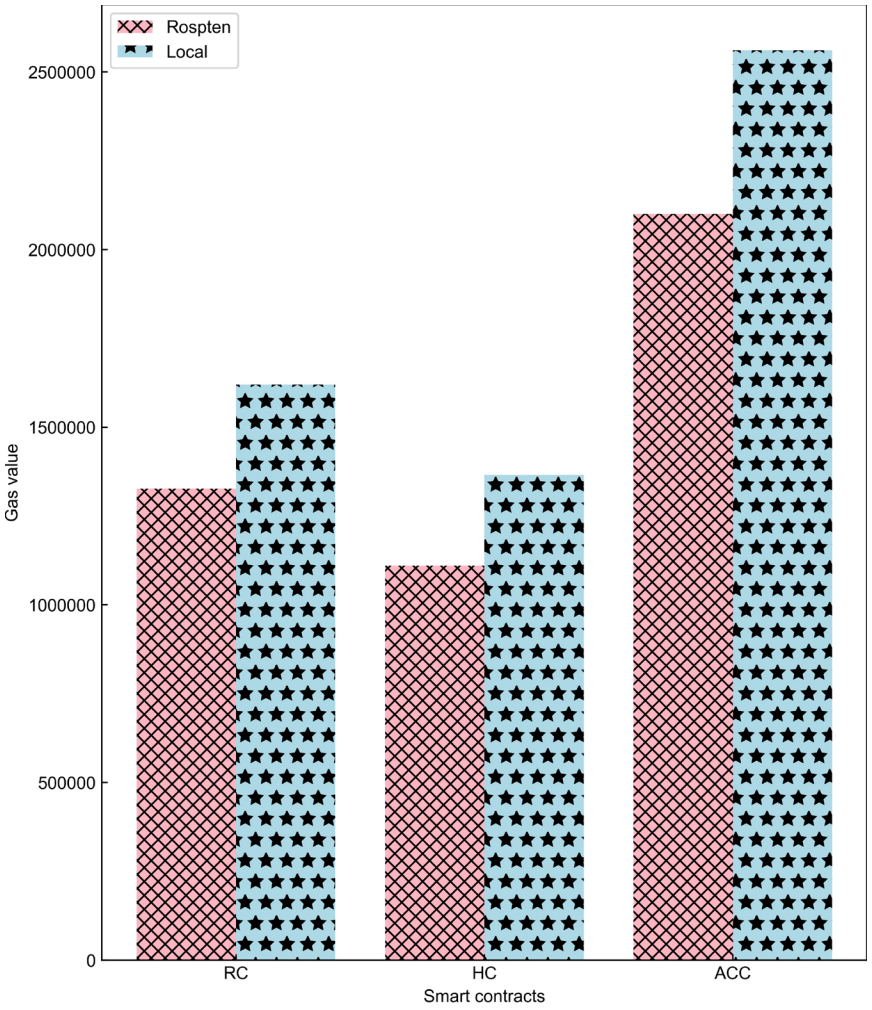

Figure 6: Gas costs of smart contracts. The x-axis represents the type of smart contract, and the y-axis represents the gas consumed by three different smart contracts

It must be said that the purpose of the experiment is to demonstrate the feasibility of the system. In summary, the experimental results in this chapter indicate that under the designed mechanism, the system can protect the security and privacy of big data, provide users with low latency services, and effectively implement data access control policies. The cost in the actual system may not be reflected in the scheme we designed. Therefore, our future work is to deploy our framework in real-world systems and proceed with extensive testing to further prove the performance of the framework.

6 Conclusion and future works

This paper was undertaken to propose a distributed privacy preservation approach according to the security and privacy problem of big data in public health emergencies, which meet the data security, privacy, and performance requirements in future networks

and systems. Our discussion began with an overview of SGX, edge computing, and smart contract, in which the basic theory and recent developments of each were briefly introduced. We then presented the system architecture which includes a smart contract framework and TEE design, combining SGX and smart contracts to build a distributed privacy protection system. Next, we discussed the framework based on the smart contract, focusing on the workflow of distributed, trustworthy access control management contracts. We then discussed the designed data access module, data protection module, and data integrity check module in TEE to achieve data security, privacy, and integrity. Finally, experiments demonstrate that the combination of SGX and smart contracts is useful, it can effectively protect the security and privacy of data and perform distributed reliable access management of data resources. However, this study was limited by the absence of SGX is attacked. Our next work will consider the study of SGX being maliciously attacked to verify how the data is affected. Notwithstanding these limitations, the study suggests that our solution is ideal for protecting data security and privacy. Moreover, the study certainly adds to our understanding of the SGX and smart contracts at a very certain level, and the findings of this study have many practical implications.

Funding Statement: This work was supported by the National Natural Science Foundation of China (Grant No. 61762033), Hainan Provincial Natural Science Foundation of China (Grant Nos. 2019RC041 and 2019RC098), Opening Project of Shanghai Trusted Industrial Control Platform (Grant No. TICPSH202003005-ZC), Ministry of Education Humanities and Social Sciences Research Program Fund Project (Grant No. 19YJA710010), and Zhejiang Public Welfare Technology Research (Grant No. LGF18F020019).

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

Aijaz, A.; Dohler, M.; Aghvami, A. H.; Friderikos, V.; Frodigh, M. (2017): Realizing the tactile internet: haptic communications over next generation 5G cellular networks. *IEEE Wireless Communications*, vol. 24, no. 2, pp. 82-89.

Anati, I.; Gueron, S.; Johnson, S. P.; Scarlata, V. R. (2013): Innovative technology for cpu based attestation and sealing. https://software.intel.com/en-us/articles/innovative-technology-for-cpu-based-attestation-and-sealing.

Azaria, A.; Ekblaw, A.; Vieira, T.; Lippman, A. (2016): MedRec: using blockchain for medical data access and permission management. *2nd International Conference on Open and Big Data*, pp. 25-30.

Bhargavan, K.; Delignat-Lavaud, A.; Fournet, C.; Gollamudi, A.; Gonthier, G. et al. (2016): Formal verification of smart contracts: short paper. *PLAS '16: Proceedings of the ACM Workshop on Programming Languages and Analysis for Security*, pp. 91-96.

Biryukov, A.; Khovratovich, D.; Tikhomirov, S. (2017): Findel: secure derivative contracts for Ethereum. *Financial Cryptography and Data Security*, vol. 10323.

Buterin, V. (2014): A next-generation smart contract and decentralized application

platform. https://cryptorating.eu/whitepapers/Ethereum/Ethereum_white_paper.pdf.

Chen, M.; Yang, J.; Hao, Y.; Mao, S.; Hwang, K. (2017): A 5G cognitive system for healthcare. *Big Data and Cognitive Computing*, vol. 1, no. 1.

Chen, Y.; Feng, Q.; Shi, W. (2018): An industrial robot system based on edge computing: an early experience. *Proceedings of Usenix Workshop on Hot Topics in Edge Computing*.

Cheng, J. R.; Li, M. Y.; Tang, X. Y.; Sheng, V. S.; Liu, Y. F. et al. (2018): Flow correlation degree optimization driven random forest for detecting DDoS attacks in cloud computing. *Security and Communication Networks*, vol. 2018.

Coronavirus (COVID-19) Data Hub (2019): *COVID-19 Activity.*

Costan, V.; Devadas, S. (2016): Intel SGX explained. *IACR Cryptology ePrint Archive*, vol. 2016, pp. 86.

Ethereum Community (2016): *An Introduction to Ethereum Platform.*

Gu, K.; Yang, L. H.; Yin, B. (2018): Location data record privacy protection based on differential privacy mechanism. *Information Technology and Control*, vol. 47, no. 4, pp. 639-654.

Hahn, A.; Singh, R.; Liu, C.; Chen, S. (2017): Smart contract-based campus demonstration of decentralized transactive energy auctions. *IEEE Power & Energy Society Innovative Smart Grid Technologies Conference*, pp. 1-5.

He, S. M.; Zeng, W. N.; Xie, K.; Yang, H. M.; Lai, M. Y. et al. (2017): PPNC: privacy preserving scheme for random linear network coding in smart grid. *KSII Transactions on Internet & Information Systems*, vol. 11, no. 3, pp. 1510-1532.

Hernández-Ramos, J. L.; Jara, A. J.; Marín, L.; Skarmeta, A. F. (2013): Distributed capability-based access control for the Internet of Things. *Journal of Internet Services and Information Security*, vol. 3, no. 3/4, pp. 1-16.

Hu, V. C.; Kuhn, D. R.; Ferraiolo, D. F.; Voas, J. (2015): Attribute-based access control. *Computer*, vol. 48, no. 2, pp. 85-88.

Intel Corporation (2015): Intel software guard extensions evaluation SDK user's guide for windows OS. https://software.intel.com/sites/products/sgx-sdk-users-guide-windows.

Intel Corporation (2016): SGX SDK. https://software.intel.com/en-us/sgx-sdk.

Intel (2016): Overview of intel protected file system library using software guard extensions. https://software.intel.com/content/www/us/en/develop/articles/overview-of-intel-protected-file-system-library-using-software-guard-extensions.html.

Jiang, X.; Coffee, M.; Bari, A.; Wang, J.; Jiang, X. et al. (2020): Towards an artificial intelligence framework for data-driven prediction of coronavirus clinical severity. *Computers, Materials & Continua*, vol. 63, no. 1, pp. 537-551.

Khalil, N. (2012): Frost & sullivan: Drowning in big data? Reducing information technology complexities and costs for healthcare organizations.

Kuhn, D. R.; Coyne, E. J.; Weil, T. R. (2010): Adding attributes to role-based access control. *Computer*, vol. 43, no. 6, pp. 79-81.

Lawless, J. (2017): "Ransomware" cyberattack cripples hospitals across England.

Li, W. J.; Chen, Z. Y.; Gao, X. Y.; Liu, W.; Wang, J. (2018): Multimodel framework

for indoor localization under mobile edge computing environment. *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4844-4853.

Mathieu, F.; Mathee, R. (2017): Blocktix: decentralized event hosting and ticket distribution network. https://blocktix.io/public/doc/blocktix-wp-draft.pdf.

McCorry, P.; Shahandashti, S. F.; Hao, F. (2017): A smart contract for boardroom voting with maximum voter privacy. *Financial Cryptography and Data Security*, vol. 10322, pp. 357-375.

Nielsen, J. (2010): Website response times. https://www.nngroup.com/articles/website-response-times/.

Notheisen, B.; Gödde, M.; Weinhardt, C. (2017): Trading stocks on blocks-engineering decentralized markets. *Lecture Notes in Computer Science*, vol. 10243.

Sargita, D.; Ankita, C.; Reshamlal, P. (2015): A review on issues and challenges of cloud computing. *International Journal of Innovations and Advancement in Computer Science*, vol. 4, pp. 81-88.

Satyanarayanan, M. (2017): The emergence of edge computing. *Computer*, vol. 50, no. 1, pp. 30-39.

Sawtooth, H. (2016): Proof of elapsed time specification.

Shi, W.; Cao, J.; Zhang, Q.; Li, Y.; Xu, L. (2016): Edge computing: vision and challenges. *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637-646.

Shi, W.; Dustdar, S. (2016): The promise of edge computing. *Computer*, vol. 49, no. 5, pp. 78-81.

Tang, Q.; Wang, K. Z.; Song, Y.; Li, F.; Park, J. H. (2019): Waiting time minimized charging and discharging strategy based on mobile edge computing supported by software defined network. *IEEE Internet of Things Journal*, pp. 1.

Wu, X. P.; Dunne, R.; Zhang, Q. Y.; Shi, W. S. (2017): Edge computing enabled smart firefighting: opportunities and challenges. *HotWeb'17: Proceedings of the Fifth ACM/IEEE Workshop on Hot Topics in Web Systems and Technologies*, no. 11, pp. 1-6.

Yi, S.; Hao, Z.; Zhang, Q.; Zhang, Q.; Shi, W. et al. (2017): Lavea: latency-aware video analytics on edge computing platform. *IEEE 37th International Conference on Distributed Computing Systems*, pp. 2573-2574.

Yin, C. Y.; Ju, X. K.; Yin, Z. C.; Wang, J. (2019): Location recommendation privacy protection method based on location sensitivity division. *Journal on Wireless Communications and Networking*, vol. 2019, no. 1, pp. 266.

Yin, C. Y.; Shi, L. F.; Sun, R. X; Wang, J. (2019): Improved collaborative filtering recommendation algorithm based on differential privacy protection. *Journal of Supercomputing*, pp. 1-14.

Yin, C. Y.; Zhou, B.; Yin, Z. C.; Wang, J. (2019): Local privacy protection classification based on human-centric computing. *Human-Centric Computing and Information Sciences*, vol. 9, no. 1, pp. 33.

Yu, W.; Liang, F.; He, X. F.; Hatcher, W. G.; Lu, C. et al. (2018): A survey on the edge computing for the internet of things. *IEEE Access*, vol. 6, pp. 6900-6919.

Zhang, F.; Cecchetti, E.; Croman, K.; Juels, A.; Shi, E. (2016): Town crier: an authenticated data feed for smart contracts. *CCS'16: Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*, pp. 270-282.