Service Chaining Security Based on Blockchain

Ligen Sun1*
1Beijing Key Laboratory of Network System Architecture and Convergence, Beijing University of Posts and Telecommunications, Beijing, 100876, China
*Corresponding author’s e-mail: Sun0610@bupt.edu.cn

Abstract. The SDN (Software Defined Networking) service chaining can take advantage of the separation of control and forwarding in the SDN network, centrally manage and coordinate various physical and virtual resources to dynamically meet users' complex network service needs. At present, most researches on SDN service chaining are about the service chaining protocol and composition structure, service chaining mapping and service chaining strategy description. However, there is no traceable data storage method for the stability of the performance of the service nodes. This paper proposes to use blockchain technology to ensure the reliable traceability of the service chaining data. Meanwhile, in order to improve the retrieval efficiency, this paper designs and implements the combination of virtual chain and Bloom filter, and through the design of smart contract to achieve the query rate of 584.7KB/s under the condition of using blockchain. Besides, the system can save 28.7% of the retrieval time in the case of a large number of service chaining data storage. These experimental results prove that the system design can be well applied in the service chaining scenario.

1. Introduction
The service chaining is a service model that passes user traffic through fixed service nodes to achieve users’ complex network service requirements. For example, in order to satisfy the HTTP/HTTPS security protection request of the user’s Web application, the user traffic is first scanned by the scanner, and then the firewall deploys corresponding protection measures to the user according to the scan result. In order to solve the DDoS attack problem faced by the user, the user’s message passes through IDS (intrusion detection system) and ADS (anti-DDoS system) in sequence. The traditional service chaining has the problems of low hardware utilization, tight coupling of network topology, and complicated deployment. The newly emerged SDN service chaining in recent years can solve these traditional problems. The SDN service chaining separates control and forwarding, and adopts virtualization technology. It has the characteristics of dynamicity, high mobility, easy to change in scale, and multi-tenancy. It can meet users' complex network service needs. The service chain we studied is the SDN service chain.

The current research work of the SDN service chaining mainly focuses on service chaining protocol and composition structure, service chaining mapping and service chaining strategy description, but there is no effective monitoring mechanism for the physical or virtual resources mastered by the SDN controller. When the services provided by organizations or individual users on cloud platforms cannot provide stable performance or even provide malicious behavior, the security of the entire service chain system will be greatly compromised. The current SDN service chain related research does not take these issues into consideration. These issues include the following parts:

1) The service chain integrates the resources of various channels, and these service nodes may provide unstable services. There is currently no traceable data storage method to collect service data to
achieve service accountability.

(2) In the service chain scenario, the service data represents the actual historical service process. These data are not allowed to be deleted or modified, and need to be stored in a way that prohibits any modification operations.

(3) Due to the complexity of the actual physical topology, man-in-the-middle attacks will cause service reports provided by service nodes to be tampered with by attackers, and a trusted data storage method that can achieve identity verification is needed.

(4) Multiple service chains in the network are interleaved at the same time. Each service chain involves different types of service nodes. It is necessary to separate the large amount of data generated and store them separately according to the service chaining to achieve traceability of service data.

In order to solve the problems mentioned above reasonably, we propose to use blockchain technology to achieve reliable traceability of service data. We chose blockchain because it can be traced and will never be tampered with, the identity cannot be stolen, and it is completely open source. Based on the above characteristics of the blockchain, we believe that the use of blockchain technology can achieve reliable and traceable service information in the service chaining, thus ensure the security.

The main contribution of our research is: 1. Based on our service chaining system[5][16] to realize the combination of blockchain, thus achieve traceability of service data. 2. Use the blockchain public-private key system and smart contract design to carry out the identity verification process to ensure the credibility of the service data. 3. Use virtual chain, Bloom filter and smart contract to achieve separate storage and quick query of service data.

The rest of the paper is structured as follows. Section II discusses related work while section III presents the system structure. The system design and implement are presented in section IV, while section V presents test results, with analyses and discussions. Finally, the conclusion is given in section VI.

2. Related works

2.1. SDN service chaining
The service chaining is a network service model that can meet the complex service logic requirements of users and pass data traffic through service nodes in a fixed order. In order to solve the problems of high hardware cost, complicated configuration and poor topology caused by the fixed structure of the traditional service chaining [1], lots of research on SDN service chaining has appeared.

Martini et al. [2] proposed an SDN controller that can implement the function of the network service chaining, which modifies the OpenFlow protocol QoS field to identify different network flow service requirements, and further allows the service sequence to be processed through a dynamically established virtual network. This SDN controller can programmatically create traffic transmission paths, thereby enabling the deployment of an adaptive network service chaining. Cerrato et al. [3] proposed to use SDN technology to dynamically reset the network paths inside the network unit. In their design, starting with high-level descriptions of the targeted graphs and the existence of a specific incident, such as connection of a new user, to the common traffic steering provided by the SDN structure, which is the whole process of dynamically instantiating network function-flow graphs (NF-FGs). Blue Planet [4] is an open source, de-commercialized service chaining orchestration platform, of which the main feature is the support of MDSO (Multi-Domain Service Orchestration). Blue Planet uses third-party SDN controllers, which can fully integrate third-party equipments and service resources. Our laboratory proposed a data-driven SDS service chaining orchestration system SDOF [5]. This system uses the SDS (Software Defined Security) [6] framework on the basis of the SDN architecture and automatically generates service chaining orchestration strategies through threat information in a data-driven manner. In this system, a unified interface for security devices is also designed and a complex Structured Threat Information Expression (STIX) [7] ontology and corresponding tools are customized to centrally collect and standardize threat information in SDS.

The above-mentioned SDN service chaining related research can decouple control plane and data
plane, thereby automatically coordinating various physical and virtual network resources to establish a dynamic network connection topology. However, these studies did not consider the impact of unstable performance nodes on the security of the entire system, and did not design a data accountability mechanism that can solve the credibility and traceability of service data. In order to solve these problems, our proposed system combines the SDN service chaining with blockchain, this scheme makes our system can be well applied in the field of service chaining data storage.

2.2. BaaS (Blockchain as a Service)

In 2008, in order to solve the credibility of digital currencies, Satoshi Nakamoto proposed the concept of blockchain technology while creating Bitcoin [8]. Because of the information traceability, identity credibility, and open application characteristics of the blockchain, it has been improved and applied to various fields, thereby solving the problem of reliable data traceability in various fields.

In IoT (Internet of Things), most of the basic equipment is highly centralized, and a centralized network infrastructure is not only prone to single point of failure, but also leads to increased end-to-end communication delays. In order to solve these problems and thus achieve the scalability and wide application of the IoT, lots of blockchain integration schemes have been proposed. Reyna et al. [9] discussed the challenges and opportunities in the research related to the combination of blockchain and IoT, and introduced different architectures of the Internet of Things based on blockchain in this paper. Liu et al. [10] proposed a blockchain-based data traceability service that does not require third-party verification. In this service, the blockchain network is a system layer that independently added to protect the traceability of data in the cloud. Users can check the loss of data integrity by sending a query request to get the record in blockchain. Biswas et al. [11] proposed a smart city solution using the blockchain's traceability and immutability features to maintain users' data. This paper suggests to use Ethereum smart contracts to achieve programmability on top of the decentralized blockchain records.

In the medical field, stored medical data relates to the security of critical treatment information, so a decentralized and tamper-resistant data storage method is needed. At the same time, an identity credibility check on the data deposit should also be implemented [12]. Some studies have combined blockchain with medical records to meet these requirements. Azaria et al. [13] developed MedRec based on blockchain smart contracts, which is a decentralized data management system. Any patient can obtain his medical information across the providers and the treatment sites. Through the properties of the blockchain, MedRec can implement identity management, confidentiality mechanisms, accountability mechanisms and data sharing mechanisms. Xia et al. [14] proposed a framework for sharing medical data between cloud service providers through blockchain contract, namely MeDShare. In order to share data securely, this article details the system design scheme. These details include system setup, requested file, package delivery, auditing and provenance, as well as each layer of the system structure and smart contract design.

3. System overview

3.1. Blockchain-based SDS service chaining system
Figure 1. blockchain-based SDS service chaining system

Figure 1 shows the architecture of our scheme in detail. The blockchain-based SDS service chaining system can automatically generate service chaining orchestration strategies and store service data in blockchain. The system consists of a five-layer structure, and these components are (from top to bottom):

**App Store Server:** This layer is the Web service layer deployed on the central server. It provides users with Web interaction services and generates orchestration strategies.

**Orchestration Engine:** The orchestration engine northbound receives the orchestration strategy and orchestration service template issued by the App Store Server, generates orchestration tasks, and schedules execution. At the same time, the generated orchestration task-related information is saved to the blockchain data storage via the blockchain API in the southbound direction.

**Blockchain Network:** The blockchain network layer stores information about orchestration tasks issued by the orchestration engine and log information of device detection results submitted by security devices. In order to make the blockchain more suitable for the application of the service chaining scenario, the design of this layer is the main work of this paper.

**Security Controller:** The security controller receives the resource invocation request of the orchestration engine through the blockchain query API, and selects the appropriate equipment from the resource pool to perform the protection task through the resource scheduling module.

**Infrastructure:** The infrastructure layer provides the required network resources for the entire security service system. These resources include network resources such as computing, storage, and security capabilities.
3.2. Blockchain-based system flow

As shown in Figure 2, the user firstly sends an orchestration strategy to the orchestration engine according to the recommendations of the App Store. After the orchestration engine receives the orchestration strategy, it combines the orchestration policy with the orchestration template provided by the App Store and the security device model abstracted from the security resources to generate orchestration jobs. At the same time, the orchestration engine will generate an order number and save it in the blockchain. When the security controller gets a new order number, it will receive the security device call request. After processing the security device call request, the security controller will invoke the corresponding security device to perform the orchestration job. When the security device finishes the security work, the result log is stored in the blockchain. The orchestration engine retrieves the device result log. If the orchestration conditions are met, it continues to call the next security device to perform the corresponding orchestration job. The next security device has the same workflow as above. Finally, when the orchestration engine determines that the service orchestration is complete, it will notify the blockchain that the order has been completed. This design can ensure that the service information in the service chain is stored in a traceable blockchain to achieve tamper-proof storage.

4. System design and implement

4.1. Ethereum and smart contract design for identity verification

The system is designed based on the Ethereum[15]. In Ethereum, the blockchain consists of a series of blocks containing smart contracts, transactions, and messages. Shown in Figure 3, a chain structure is formed by saving the hash value of the previous block in every block. Among them, the transaction is used to initiate the transfer of virtual currency, and the smart contract is automatically executed without the supervision of any third-party organizations. By initiating a transaction to a smart contract, the state of the internal variables of the smart contract can be changed.
In the service chaining scenario, we do not initiate transactions to transfer virtual currency to users, but use it to store service data. Hence for the identity verification, we carried out a blockchain transformation: register the identities of all the devices in the smart contract, and when a device wants to save the data to the system, the system uses the public/private key in the blockchain for verification. Although this design does not involve virtual currency rewards, the malicious information sent by the device will be punished by the service accountability.

![Identity Management Smart Contract](image)

**Figure 4. Identity management smart contract**

We design and implement an identity management smart contract to save the public key information of all registered devices. As shown in Figure 4, when these registered devices send data to the blockchain, they encrypt the service data with their private key and attach their public key. At the same time, the blockchain nodes query and verify the identity of this device, and only stores legal information in the blockchain. In this way, man-in-the-middle attacks will be difficult to implement for the reason that the attacker cannot obtain the device's private key, so the credibility is guaranteed.

4.2. **Smart contract design for service data storage**

![Storage Smart Contract](image)

**Figure 5. Storage smart contract**

The storage smart contract as shown in Figure 5 is divided into two parts, a temporary storage area and a historical data area. The temporary storage area stores the log data location of the service chaining that is being executed, and the historical data area stores the historical log data of the service chaining that has been executed. The data in the temporary storage area is stored in the order ID->pointer mapping relationship, where the order ID represents the order number of the service chaining being executed, and
the pointer represents the current data storage location. When new information is being stored, the information submitter saves the old pointer to the data and sends the old pointer and the new pointer to the smart contract. After receiving the information, the smart contract updates the new pointer corresponding to the order ID. The orchestration engine will notify the smart contract after confirming the completion of the service chaining. Then smart contract places the order information in the temporary storage area into the last list in the historical data area and updates the Bloom filter of that list. When the length of the list reaches the maximum limit length, the smart contract will automatically create a new list and initialize a bloom filter which is all 0.

4.3. Bloom filter

Traditional information retrieval methods are performed by changing time for space or space for time. Bloom filters break out of this cycle by introducing a new variable: the error rate. This transformation reduces both time and space complexity to $O(1)$. And error rate can be equivalent to a small amount of time consumption expectation in the service chaining scenario, which is completely acceptable.

![Figure 6. Bloom filter](image)

Figure 6. Bloom filter

Shown in Figure 6, bloom filter is a bit array of length $m$, which is initialized to all 0 in the beginning. When the data $A$ is mapped to the Bloom filter, we select $k$ hash functions to calculate the $k$ hash values of $A$, and modulo these $k$ results by $m$ to obtain the index value in the bloom filter. The values corresponding to these index values in the bit array then be modified to 1. The same is true for data $B$.

When detecting whether data $C$ is included in the set, obtain $k$ index values through the above steps first. Check whether the corresponding positions in the bit array are 1. If not all the positions are 1, the data $C$ is not in the bloom filter. Obviously data $C$ above is not included in the bloom filter.

4.4. Virtual chain and quick query

![Figure 7. Virtual chain design](image)

Figure 7. Virtual chain design

The structural characteristics of the blockchain determine that all data is stored on the same chain. When data is retrieved, it is necessary to find the required information from beginning to end. Although the current computer's computing power combined with regular matching is sufficient to complete such
work, as the amount of data increases, this method will consume a lot of resources and waste lots time. In order to achieve fast retrieval, this paper studies and implements the logical function of virtual chaining.

As shown in the figure 7, the data of multiple service chainings in the blockchain network are mixed. The virtual chaining is implemented by recording the transaction hash of the previous data on the same service chaining in new data. In this way, a virtual chaining can be generated through hash transfer to achieve the purpose of fast retrieval.

The quick retrieval of service orchestration log information is divided into two parts. The first part is the retrieval of log information for the currently unfinished service chaining, and the other is the retrieval of completed historical service log information. For the current unfinished service chaining, its data is stored in the temporary storage area of the smart contract. The current latest data pointer can be obtained by directly searching the order ID. By indexing forward through this pointer, you can find all the log data of this service chaining. For the completed historical service chaining data, you need to use hash functions and modulo functions to calculate k index values of order ID in the bloom filter, and then use this data to compare all of the historical service chaining data lists’ bloom filters. When you find the correct list, you can further search for the order.

5. Test results and analysis

5.1. Test design
In order to verify that the design we proposed can improve the efficiency of data retrieval, and thus is more suitable for the data traceability in the service chaining scenario, we conducted experiments based on the real Ethereum environment to test the virtual chain technology and the Bloom filter on retrieval efficiency. In the experiment, we assume that each service chaining data is composed of data stored in 5 blockchain transactions, and each transaction contains about 3KB of log data (normally distributed with 3KB as the center). And DAG (Directed Acyclic Graph) is used to provide parallel storage of the blockchain to improve the throughput of the blockchain. Our experiment consists of three parts: First, we tested the time required for various activities to interact with the blockchain using the web3.py plugin (including initializing interactive tools, querying block information, querying transaction information, and query information from smart contracts), through which we can intuitively observe the time cost of using the blockchain. Then we compared the difference in the efficiency of retrieving effective information between the virtual chain and the traditional method of querying blocks. Finally, we tested the time of retrieving with the increase of the total number of service chainings with the addition of the Bloom filter and without the Bloom filter.

5.2. Time consumption of operations
At first, we communicated with the Ethereum private chain Geth through the web3.py script, and detected the time required for each operation. This work is to compare the time required for different operations, so that the time cost of blockchain data operations can be more intuitively evaluated.

| Table 1. Time consumption in web3.py |
|--------------------------------------|
| Operation    | Initialize Web3 | Get Block | Get Transaction | Interacting With Smart Contract |
| Time Consumption (ms) | 0.726938 | 2.586758 | 3.003525 | 9.911728 |

We draw the conclusions through the Table 1: when using python scripts to interact with the Ethereum blockchain, the time consumed by these blockchain data operations is relatively low, at most less than 10ms. These data are determined by the physical distance between the blockchain node and the query user and the network topology. If the closest blockchain node that the user can access is far from
the user, the time consumption of these data operations will be larger and vice versa. At the same time, we observe that the time consumption required to initialize the web3 script is only 0.73ms, indicating that interacting with the blockchain is a very convenient way of information retrieval. Using the blockchain is suitable for data retrieval in the service chaining. We also found that the time required to interact with smart contracts is much longer than the time used to retrieve blocks and transactions, which shows that the process of using smart contracts is more complicated. However, in our design, only a limited number of calls are required for smart contract.

5.3. Efficiency of virtual chain
In order to prove that the use of virtual chain can separate service chaining data from the physical structure of the blockchain by using logical relationships, thus improve retrieval efficiency in a service chaining scenario, we compared the effective information obtained every second with and without the virtual chain. In this experiment, we still used the experimental environment mentioned above. In the experiment, the traditional method without adding a virtual chain used the method of querying blocks to find data information related to the service chain ID. And the experiments were conducted under the delay caused by the network topology $u = 0$ms (under ideal conditions), 0.5ms and 1ms respectively.

![Figure 8. Effective data received per second as the service node logs increase](image)

Through the result from Figure 8, we obtained these conclusions: due to the influence of physical location and network topology, the rate of retrieving effective information is decreasing as the network delay increases, and the impact of network delay on block query method is larger (1ms delay in 200 logs can reduce the retrieval rate of block query by 60.9%, while the virtual chain only receives 31.6% impact under the same condition). When a small amount of data is stored in the blockchain, the traditional method of querying the block to retrieve information from the blockchain has higher efficiency than the method of using the virtual chain, but as the amount of data continues to increase, the rate of traditional method will decrease in the form of a logarithmic function, and even if there is no delay in 1000 logs, it is only 57.9KB/s. In contrast, the retrieval rate using the virtual chain is always maintained at a high level, even with a delay of 1ms, there is a stable rate of 399.5KB/s, and without delay, it can reach 584.7KB/s. This is because with the increase of service node log information, the method of querying blocks requires obtaining a large amount of data to obtain effective service chaining information. In this process, a large number of data operations to obtain blocks will be executed. In contrast, after virtual chain queries a smart contract, it will purposefully find the transaction which store the service chain data according to the data pointer, and its time complexity is only depend on the number of effective transactions. In the service chain scenario, especially when users need to frequently obtain service chain data, the effective information retrieval rate of 57.9KB/s or even lower obviously
cannot meet the demand, and as the amount of data increases, the lower retrieval rate will make the system worse. The stable 584.7KB/s of virtual chain can obviously retrieve a large amount of effective data in the blockchain network to meet the needs of users. Through the above analysis, the use of virtual chain technology can make the blockchain more suitable for the scenario where a large amount of data is staggered in the service chaining.

5.4. Bloom filter’s performance

The Bloom filter is used in many retrieval scenarios with a large amount of data due to its retrieval time complexity of $O(1)$. In order to verify whether the Bloom filter can be used in the service chain scenario and to explore the conditions of its use, we compared the retrieval time using the Bloom filter and not using it when the number of service chainings increases. In this experiment, we selected 50 different hash functions for hash calculation, the corresponding Bloom filter array length is also 50.

![Figure 9. Retrieval time as the number of service chainings increases](image)

We draw the following conclusions from the Figure 9: First, regardless of whether Bloom filter is used, the retrieval time of both retrieval methods changes linearly with the increase of the number of service chainings, which is consistent with the retrieval time changing experimental expectations. At the same time, we found that these two polylines are not strictly straight lines, this is due to the random distribution of the required retrieval data. We took the average of the time required for 20 searches to make the graph. This image can be closer to the expected experimental result of a straight line as the number of experiments increases. We also found that using the Bloom filter can partially reduce the time required to retrieve the service chaining when there are a large number of service chains. When there are 300,000 service chains, it can save approximately 28.7% of the retrieval time. When the number of service chains is small (less than 70,000 service chains), it is not suitable to use the Bloom filter for retrieval. This is because the Bloom filter takes time to perform hash calculation and value comparison. When the service data is little these calculation operations will cause more retrieval time consumption than ordinary retrieval method. In the service chaining scenario, you can choose whether to use the Bloom filter to improve retrieval efficiency according to the amount of data. In addition, we found that the use of smart contracts in the scenario of 300,000 service chainings only requires 12.2ms, and the system design we have proposed can better meet the user’s needs for data retrieval in the scenario of service chaining.

6. Conclusions

In response to the problem that service nodes in service chaining may provide unstable services, we have designed and implemented a system based on blockchain technology to achieve reliable and traceable
service data in service chaining scenarios. In our design, in order to realize the management of the service chaining information, we use smart contracts to store the service chaining ID and trusted publisher identity information. In terms of improving retrieval efficiency, we found that the use of virtual chain technology can maintain a stable retrieval rate of 584.7KB/s when the amount of data increases, while the querying blocks method can only achieve a retrieval rate of 59.7KB/s when there are 1,000 service data. We also conducted experiments to verify that when storing more than 70,000 service chains, using the Bloom filter can partially reduce the time to retrieve the service chaining (around 28.7% less time), and as the number of service chaining increases, the proportion of time saved increases. Through the experimental results above, the system we proposed can effectively ensure the credible traceability of data in the service chaining scenario, thereby achieving accountability for some unstable service behaviors or even malicious service behaviors. The system design we proposed can be applied to the service chaining that integrates a large number of cloud platform resources, especially the physical or virtual device resources provided by individuals and organizations. This design can avoid malicious users from damaging the security of the entire service chaining system. At the same time, we also found that the Ethereum smart contract design implemented in the system provides blockchain-based programmability for the service. Using smart contracts can make the blockchain suitable for data traceability in various scenarios, and unique blockchain designs in different fields can be customized by integrating various algorithms in smart contracts. Last but not least, the system we designed is based on the decentralized nature of the blockchain, which can also reduce the occurrence of single point of failure in the network.

In the future, we will continue to think about the security of information in the system, add access control functions to protect the privacy of user data. As for storage of data, we will further study the unified data storage format to standardize system processes.

Acknowledgments
This work was supported by Networks National High Technology Research and Development Program 863: trusted and controllable technology and its supporting system of cloud computing platform (NO.2015AA016003) and Beijing Laboratory of Advanced Information Networks.

References
[1] Quinn, P., Nadeau, T. (2015) Problem Statement for Service Function Chaining. Ietf Rfc., 53: 1689-1699.
[2] Martini, B., Paganelli, F., Mohammed, A.A., Gharbaoui, M., Sgambelluri, A., Castoldi, P. SDN controller for context-aware data delivery in dynamic service chaining. In: 2015 1st IEEE Conference on Network Softwareization (NetSoft). London. pp. 1-5.
[3] Cerrato, I., Jungel, T., Palesandro, A., Risso, F., Suñé, M., Woesner, H. (2014) User-Specific Network Service Functions in an SDN-enabled Network Node. In: 2014 Third European Workshop on Software Defined Networks. London. pp. 135-136.
[4] Blueplanet. (2015) Multi-domain service orchestration. http://www.blueplanet.com/products/multi-domain-service-orchestration.html.
[5] Wang, W., Qiu, X., Sun, L., Zhao, R. (2016) A Data Driven Orchestration Framework in Software Defined Security. In: IEEE International Conference on Network Infrastructure and Digital Content. Beijing,China. pp. 34-39.
[6] Liang, X., Qiu, X. (2016) A software defined security architecture for SDN-based 5G network. In: 2016 IEEE International Conference on Network Infrastructure and Digital Content (IC-NIDC). Beijing,China. pp. 17-21.
[7] Barnum, B. (2012) Standardizing cyber threat intelligence information with the structured threat information expression (stix). MITRE Corporation., 11: 1–22.
[8] Nakamoto, S. (2008) Bitcoin: A peer-to-peer electronic cash system. https://bitcoin.org/bitcoin.pdf.
[9] Reyna, A., Martín, C., Chen, J., Soler, E., Díaz, M. (2018). On blockchain and its integration with iot. challenges and opportunities. Future generation computer systems., 88: 173-190.
[10] Liu, B., Yu, X.L., Chen, S., Xu, X., Zhu, L. (2017). Blockchain Based Data Integrity Service Framework for IoT Data. In: IEEE International Conference on Web Services (ICWS). Honolulu, HI. pp. 468–475.

[11] Biswas, K., Muthukumarasamy, V. (2016) Securing smart cities using blockchain technology. In: IEEE 18th International Conference on High Performance Computing and Communications; IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems (HPCC/SmartCity/DSS). Sydney, NSW. pp. 1392–1393.

[12] Mettler, M. (2016) Blockchain technology in healthcare: The revolution starts here. In: 2016 IEEE 18th International Conference on e-Health Networking, Applications and Services (Healthcom). Munich. pp. 1-3.

[13] Azaria, A., Ekblaw, A., Vieira, T., Lippman, A. (2016) MedRec: Using blockchain for medical data access and permission management. In: 2016 IEEE 2nd International Conference on Open and Big Data (OBD). Vienna. pp. 25-30.

[14] Xia, Q., Sifah, E.B., Asamoah, K.O., Gao, J., Du, X., Guizani, M. (2017) MeDShare: Trust-less medical data sharing among cloud service providers via blockchain. IEEE Access., 5: 14757–14767.

[15] Wohrer, M., Zdun, U. (2018) Smart contracts: security patterns in the ethereum ecosystem and solidity. In: 2018 International Workshop on Blockchain Oriented Software Engineering (IWBOSE). Campobasso, Italy. pp. 2-8.

[16] Zhang, G., Qiu, X., Gao, Y. (2019) Software Defined Security Architecture with Deep Learning-Based Network Anomaly Detection Module. In: 2019 IEEE 11th International Conference on Communication Software and Networks (ICCSN). Chongqing, China. pp. 784-788.