FRChain: A Blockchain-based Flow-Rules-oriented Data Forwarding Security Scheme in SDN

Weichen Lian, Zhaobin Li *, Chao Guo, Zhanzhen Wei, and Xingyuan Peng
Department of Electronics and Communication Engineering
Beijing Electronic Science & Technology Institute
Beijing, 100070, China
[e-mail: kevinbaylor@163.com, bestibesti@163.com guo99chao@163.com, 4193026@qq.com, 19801277223@163.com]
*Corresponding author: Zhaobin Li

Received July 7, 2020; revised November 9, 2020; accepted January 14, 2021; published January 31, 2021

Abstract

As the next-generation network architecture, software-defined networking (SDN) has great potential. But how to forward data packets safely is a big challenge today. In SDN, packets are transferred according to flow rules which are made and delivered by the controller. Once flow rules are modified, the packets might be redirected or dropped. According to related research, we believe that the key to forward data flows safely is keeping the consistency of flow rules. However, existing solutions place little emphasis on the safety of flow rules. After summarizing the shortcomings of the existing solutions, we propose FRChain to ensure the security of SDN data forwarding. FRChain is a novel scheme that uses blockchain to secure flow rules in SDN and to detect compromised nodes in the network when the proportion of malicious nodes is less than one-third. The scheme places the flow strategies into blockchain in form of transactions. Once an unmatched flow rule is detected, the system will issue the problem by initiating a vote and possible attacks will be deduced based on the results. To simulate the scheme, we utilize BigchainDB, which has good performance in data processing, to handle transactions. The experimental results show that the scheme is feasible, and the additional overhead for network performance and system performance is less than similar solutions. Overall, FRChain can detect suspicious behaviors and deduce malicious nodes to keep the consistency of flow rules in SDN.

Keywords: SDN, Data Forwarding Security, Blockchain, Flow Rules, BigchainDB

This paper was supported by the Fundamental Scientific Research Business Funds for the Central Universities "Basic Scientific Research Business Fee" (Project Number: 328201911).

http://doi.org/10.3837/tiis.2021.01.015 **ISSN : 1976-7277**

1. Introduction

With the development of SDN, more security problems appear [1]. There are two main types of attack: one is to cause bad impact on normal operation of network devices functionally, such as DDoS attack [2], topology poisoning attack [3]. The other is to monitor, tamper, forge, or hijack data packets. The latter one is hard to detect and could cause tremendous trouble without being noticed.

In the data forwarding process of OpenFlow networks [4], controllers map out strategies and deliver them to flow tables in switches through TCP/TLS channels. During the process, three types of trouble could appear: 1) hijacked or unauthorized controllers could make and deliver wrong flow rules [5]. 2) Man-in-the-Middle attack (MitM) might be performed even though the controllers and switches are connected by LLS/TLS channels [6]. 3) Compromised or unauthorized switches might forward packets in wrong ways [7]. Currently, there are three types of solutions to solve these problems: design secure controllers, monitor the data forwarding process, and ensure the consistency of flow rules.

SDN is a typical flow-driven network. The consistency of flow rules are the basic elements to ensure the efficient operation of SDN. Due to the forwarding devices fully trust the flow rules, once the flow rules that are provided by the fake controller or the malicious application are executed, the switch will fail to handle data packets correctly. Therefore, ensuring the consistency of the flow rules, preventing the proliferation of malicious strategies and ensuring the correct delivery and execution of various flow rules are critical to the safe operation of SDN [8-10]. But in current SDN related research, there are few traffic policy security solutions.

In this paper, we propose FRChain, a blockchain-based data secure forwarding scheme in SDN. We use the blockchain to preserve the flow rules in SDN. We also designed a voting process to ensure the correctness of the flow rules in case of conflicts between different nodes. Furthermore, FRChain aims to guarantee the consistency of the SDN traffic policy by using different blockchain solutions in both single-controller and multi-controller scenarios. In the last part, we implement FRChain by using Ryu controller, Open vSwitch, and BigchainDB. We also made performance tests and compared it with similar solutions

Our contributions. The contribution of this paper is listed as follows:

- We studied related data forwarding schemes in SDN and found out that the core of ensuring forwarding security is keeping the consistency of flow rules.
- We propose a secure forwarding scheme called FRChain which is the first to using blockchain to solve data forwarding security problem in SDN:
  - Devices including controllers and switches in the same SDN network are collaborators of blockchain network. However, only the controllers have the privilege to post transactions whose payload is flow strategies.
  - By regularly comparing the flow rules in the local flow tables of switches with the flow rules in the blockchain to determine whether the strategies have been tampered with.
  - If an inconsistent flow rule is detected, the switch that discovered the problem makes a voting request. Other nodes except the controller will perform their own verification and vote based on the result which is a reference of the conclusion.
  - Merkle tree is used to calculate the root hash of the invalid flow rules to reduce storage consumption.

- To avoid the confliction between the SDN centralized architecture and the decentralized structure of the blockchain, we use consortium chain and private chain, instead of public chain, under multi-controller and single-controller scenarios respectively, which ensures that the controller is still the control center of the network.
- We implemented the FRChain in software-based switch, and conducted functional and performance tests. The test results show that this solution will increase the system overhead accordingly. But when the number of flow rules processed reaches a certain number, the system overhead tends to be stable, with a CPU utilization rate of approximately 6% and a memory utilization rate of approximately 10%. We believe that FRChain can be deployed on small-scale networks to ensure the security of data forwarding, such as data exchange between enterprises.

The rest of the paper is organized as follows. Section 2 presents the background and related work of data forwarding security problem in SDN. Section 3 illustrates the design and implementation. Section 4 describes the evaluation of FRChain and the test results. At last, Section 5 shows the conclusion and future work.

2. Background and Related Work

This section mainly introduces related research and solutions of the scheme. Firstly, we'll address the premise of FRChain and introduce the security problems of the Control Plane, the Southbound interface and the Data Plane. Secondly, we'll illustrate some solutions in securing the data forwarding process in SDN. Thirdly, some related research of blockchain will be shown in this part. Lastly, we will introduce the BigchainDB and explain why we use it as a security measure.

2.1 Data Forwarding Related Threats and Premise in SDN

Generally, the data forwarding process in SDN is consists of three parts, the Control Plane, Southbound API, and the Data Plane. 1)As the most important component of the Control Plane, the controller is mainly used to regulate the network. 2)In the Data Plane, the OpenFlow switch obligates to process data packets directly. 3)The Southbound API refers to the OpenFlow channels between the Control Plane and the Data Plane. In this subsection, we'll discuss the vulnerability of these three parts respectively.

2.1.1 The Control Plane

The controller is the most important part in the Control Plane for its global view and centralized dispatcher function. It is an application that conducts switches via delivering forwarding policies which is called flow rules to realize centralized control of data flow indirectly. The feature has brought convenience for administrators to perform new protocols but it also led to probability of single point failure. Once a controller is attacked or compromised, the operation of network will be posed into threats.

In this paper, we presuppose that the controller is safe so that the flow rules will thereby be correct. If not, once a controller is malicious, the network will break down and there will be no chance to secure any data.

2.1.2 The Data Plane

Open vSwitch (OVS) is the most popular software-based switch in the data plane and it is open-source for virtual switching research and experimentation. Flow tables that are stored in the kernel space of OVS can provide guidance for forwarding packets.

One of the most destructive methods of striking is the MitM attack that attackers can create independent contacts with both ends of the communication. It'll make the sending and receiving parties mistakenly think that they are communicating directly with each other. But in fact, the whole session is completely controlled by the attacker. In this paper, we propose a threat model to perform MitM attack and explain how FRChain detect the behavior.

2.1.3 The Southbound API.

The Southbound API refers to the OpenFlow channels between the switch and the controller. It is mainly used for the controller to detect the status of the switches and ensure that it can issue instructions as required. The OpenFlow channel can be either an unencrypted TCP connection or an encrypted TLS connection. On one hand, using TCP connections will increase the network performance but will reduce the security level. On the other hand, although TLS connections will make it reliable, it still can't resist the MitM attack. Supposing that the OpenFlow channel is compromised, the attackers can forge flow entries and thereby control the switches in the network or even shut down all connections.

In centralized solutions, for example, if we appoint controller as the root of trust, the risk we mentioned above still exists because it can't defend the MitM attack. Therefore, FRChain provides a distributed scheme to prevent such a situation that one or more OpenFlow channels are compromised by attackers by sharing a consensus network.

2.2 Data Forwarding Security Relevant Research in SDN

Forwarding data packets safely is the premise of the operation of an SDN-enabled network. For now, there are mainly three categories of solutions to solve the problems: design and develop new security controllers, data packet security technologies, and detect the legality and consistency of flow rules.

Al-Shaer et al. [11] re-encoded the configuration information of the OpenFlow flow table by binary decision graph technology and combined with model detection technology, proposed the FlowChecker detection system. Son et al. [12] proposed the FLOVER model detection system in combination with the Yices SMT solver, which converts OpenFlow's flow rules and network security-related policies into corresponding assertions, to detect whether flow rules match the analysis result given by the SMT solver security policy. Khurshid et al. [13] proposed the VeriFlow detection model for legality detection of flow rules, to check the violations across the network and locate and handle the violation flow rules in real-time. Although these methods can guarantee the consistency of the flow rules by perceiving the changes of the flow tables dynamically, it cannot defend the MitM attacks in the OpenFlow channels between controllers and switches. Also, the implementation of the above solutions is relatively complex, and the new or improved functions also bring a greater burden to the controllers. Chi et al. [14] analyzed several cases in which the switches were maliciously compromised and proposed the use of a probe packet to discover the malicious switch. Zuo Zhibin et al. [15] utilized P4 forwarding devices to parse the password identification that was added to the data packet. But these solutions may have two potential threats. One is the packet loss caused by unstable network conditions. The other is that they cannot prevent attackers from replicating data flows by modifying the flow tables. Qiu et al. [16] realized the global flow table by real-time access to the whole network topology and flow information to monitor

network behavior. However, this solution is unable to cover all forwarding devices and flow rules especially when the switches are deployed at the edge of the network and this scheme can't detect the modified flow entries. Sasaki T et al. [17] proposed SDNsec to monitor the data forwarding process by embedding encrypted labels in packets header. Although the scheme can provide strong protection for data packets, it will increase much extra burden in the forwarding devices.

Although the above solutions can solve the problem to a certain extent, the program itself still has some shortcomings. We believe that the key to solving data forwarding security problem lies in how to protect flow entries from being modified, deleted and added without permission. Therefore, we propose FRChain that uses blockchain to keep data flow safe.

2.3 Related Applications of Blockchain and SDN

Blockchain was originally proposed as the underlying technology for Bitcoin, first proposed in [18]. Now it has been used widely in the security field due to its distributed storage, decentralized management, and difficulties in tampering information. With the development of blockchain, the digital encrypted currency is no longer the only option. It can be used in more aspects, like securing important data.

In many frontier areas like the Internet of Things (IoT) and Cloud Computing, blockchain is used to keep important information safe, including authorization information and certification tickets, etc. Kataoka et al. [19] used blockchain to store IoT device authentication information that has been authenticated by the edge gateway devices. Tselios et al. [20] proposed using blockchain to synchronize IoT messages across network-wide devices. Ali et al. [21] used external datastores for actual bulk storage to implement PKI. Sharma et al. [22] used blockchain to share service provider information in a distributed cloud architecture with an SDN-enabled networking. Sharma et al. [23] proposed DistBlockNet which used blockchain to secure SDN architecture for IoT networks. Chen et al. [24] designed a distributed ledger for peer-to-peer IoT networks by using blockchain. Singh et al. [25] utilized blockchain to provide a trustworthy data-sharing environment for the Intelligent vehicle. Yazdinejad et al. [26] proposed an authentication approach that utilizes blockchain and SDN to ensure low latency in 5G network by reducing unnecessary re-authentication. Houda Z A E et al. [27] proposed Cochain-SC to defend DDoS attack by sharing malicious IP address in blockchain. Besides, blockchain has been used in some other frontiers to keep valuable data safe. Zhang et al. [28] proposed a data security sharing and storage system named DSSCB that is based on consortium blockchain. Kang J et al. [29] proposed a credential-based data sharing scheme to store and share data in the vehicle edge network by using blockchain and smart contract.

According to the existing application scenarios and solutions, blockchain is mainly used to store important information. The designers put their trust in a distributed architecture and consensus networking to reduce the risk of single point failure. In SDN, devices in data plane handle packets under the instructions of the flow tables. We treat the flow entry as a sticking point and we therefore propose to utilize blockchain to preserve the flow rules.

2.4 How to Choose Blockchain

With the gradual enrichment of network service types and the rapid increase of network traffic, more flow instructions will be needed in SDN. When processing large amounts of data, traditional blockchain may perform poorly in data storage performance. Thus, we choose BigchainDB to implement the scheme. The advantages of BigchainDB are listed as follows.

(1) No Need to Prove the Workload. The consensus algorithm in the blockchain enables all nodes to share the same premise in jointly dealing with a certain stuff. The earliest consensus algorithm is the proof of work (POW), which was the basis of the Bitcoin and the Ethereum. Most of the solutions above intended to choose the well-developed Ethereum and

use the POW as the consensus algorithm. However, it has to be mentioned that using Ethereum means the system has to pay for each transaction, which will lead to extra overhead. Therefore, we need some other blockchain to avoid the extra costs.

The BigchainDB is based on the Tendermint consensus algorithm. The algorithm resembles the Practical Byzantine Fault Tolerance (pBFT) algorithm and both of them need at least four nodes to build the network. Let's assume the amount of malicious node is m and the total number of nodes is T. Under the Tendermint algorithm, a relation is required to ensure that the system operates normally in an untrusted environment:

$$|T| \geq 3m + 1(t \geq 1) \tag{1}$$

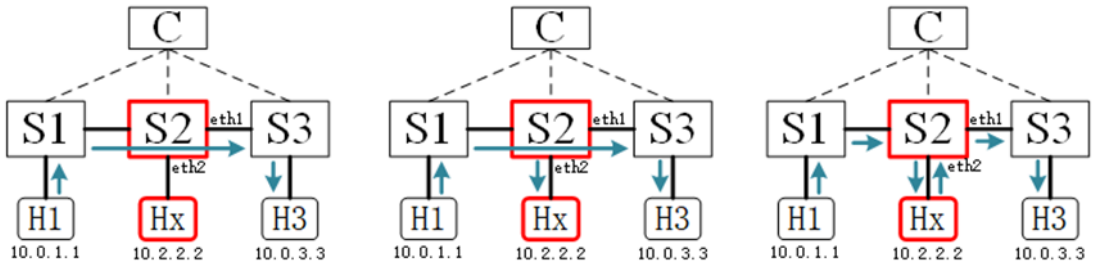
Thus, the minimum value of R is four and our system thereby requires at least four nodes.

(2) Outstanding Performance in data storage. BigchainDB is a decentralized database and it points to the performance of 1 million writes per second throughput, storing petabytes of data, and sub-second latency [30]. The high performance of BigchainDB makes it convenient to store numerous data, and its characteristics improve the safety and invariability of the storage. The underlying component of BigchainDB for storing data is MongoDB, which is a distributed database.

When dealing with massive data, traditional blockchain and distributed database can either behave bad in performance or be short of consensus. However, BigchainDB has good potential to handle massive flow entries and keep the consensus network. So BigchainDB is a suitable component for FRChain.

2.5 Threat Model

During the data forwarding process, packets might be eavesdropped, redirected, modified or even be dropped. In this subsection, we'll address a threat model that is fairly common and easy-to-implement in SDN. We choose a basic SDN single-controller linear topology with three switches to interpret how attacks are performed. Let's assume that the attacker has already got system privileges and he thereby can control the software-based switch and perform arbitrary operations on the flow table.



Src_ip	Dst_ip	Action
10.0.1.1	10.0.3.3	- Forward to eth1

(a) Benign flow entry of *S2*

Src_ip	Dst_ip	Action
10.0.1.1	10.0.3.3	- Forward to eth1 ***- Set dst_ip=10.2.2.2, Forward to eth2***

(b)The Eavesdropping attack

Src_ip	Dst_ip	Action
10.0.1.1	10.0.3.3	***- Set dst_ip=10.2.2.2, Forward to eth2***
10.2.2.2	10.0.3.3	***- Set src_ip=10.0.1.1, Forward to eth1***

(c)The MitM attack

Fig. 1. How the Compromised S2 Performs Eavesdropping Attack and Man-in-the-Middle Attack

As shown in **Fig. 1**, *S2* is a compromised switch with an evil host *Hx* connected to port *eth2*. If H1 wants to send messages to *H3* through these three switches, a benign flow entry will be delivered to *S2* as shown in **Fig. 1(a)**.

- ***The eavesdropping attack.*** As shown in **Fig. 1(b)**, the attacker can grab the packets by adding a new action in the original flow entry. Then the flow will be sent to both *H3* and *Hx* through ports *eth1* and *eth2* separately. As a result, the attacker can wiretap all packets from *H1* without being noticed.
- ***The Man-in-the-Middle attack.*** To perform a Man-in-the-Middle attack, as shown in **Fig. 1(c)**, the attacker can intercept the packets by changing the destination port to *eth2* and converting the destination IP into his address when the data flow passes the compromised switch *S2*. After that, *S2* may be added a new flow entry to send the intercepted packets back to *H3* through port *eth1* and set its source IP address to *H1* so that the attack behavior will be covered.

In this section, we introduce related research of this paper and propose a threat model. In the area of data forwarding security, the defensive scope of the existing solutions either fails to cover all flow rules or gathers all flow rules on a single point. In this case, once the safety-related node fails, the solution cannot continue to execute. In addition, these solutions lack of effective source tracing capabilities for the varieties of flow entries, making it difficult to locate devices with potential security risks. Therefore, we proposed FRChain which use blockchain to avoid the paralysis of system security functions caused by single-point defects in an untrusted environment, and the protection scope simultaneously covers all flow rules. Moreover, it can also increase the traceability of attack behaviors in SDN.

3. Proposed Scheme Design

In this section, we will specifically depict our design. Part 3.1 shows the global view of our scheme. After that, the working process will be illustrated in Part 3.2. Then we'll explain the design of our scheme theoretically in Part 3.3. At last, we will analyze the security of our design in Part 3.4.

3.1 Basis of the Scheme

According to the size of SDN-enabled network cluster, it can be mainly divided into two application scenarios: the single-controller network and the multi-controller network. To avoid the logical conflict between the centralized structure of SDN and the decentralized characteristics of blockchain, we apply the private chain and the consortium chain in our scheme instead of the public chain. In FRChain, the single-controller network can use a private blockchain, while the multi-controller can use a consortium blockchain. In order that we can assign permissions to devices accordingly. Flow tables are the guidance for switches to process the data packets, which are made by and delivered from controllers.

In FRChain, as shown in **Fig. 2**, every device except hosts in an SDN network is selected as a node and all devices share the same blockchain. Only the controllers have the right to post the transactions, while others can only read the contents. A new flow entry will be delivered to switches in two ways: 1) the OpenFlow channels and 2) the synchronous blockchain network.

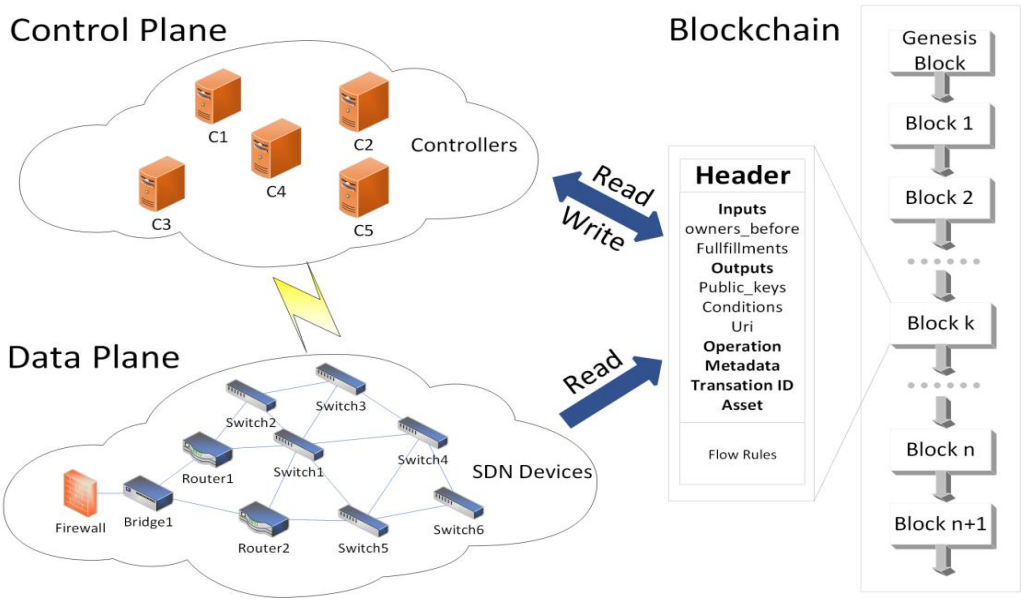


Fig. 2. Overview of FRChain

3.2 Working Process

In this part, we take a topology with one controller and four switches as an example to illustrate the process of FRChain. As shown in **Fig. 3**, Host A and Host B connect to switch 1 and switch 3 respectively. We assume Host A wants to send messages to Host B.

1) S1 receives a message from Host A and then checks the local flow tables to find matched flow entries. If nothing matches, a packet-in message will be posted to the controller for instructions.
2) After receiving the packet-in message, the controller makes and delivers the flow rules to all devices in path. Simultaneously, the controller posts transactions whose assets are key information of new flow strategies into the blockchain. Then the controller sends out the new flow rules in forms of packet-out messages.
3) All switches will follow the new instructions to forward the packets. Occasionally, each switch needs to compare the key values extracted from local flow tables and the assets in blockchain. Let's assume an unmatched item is detected in switch 1, this node will broadcast a ticket and call for a vote.
4) Other nodes will receive the request and the ticket. Each node except the controller will start a new thread to do the comparation and vote according to the results respectively. The controller will take corresponding measures based on the voting result.
5) The vote proposer S1 will be demanded to prove that the conflict is caused by a benign flow entry. It can broadcast the ticket that is signed by S1's private key into the blockchain. Then the controller will verify the signed ticket by S1's public key. If it can pass the verification, the controller will thereby trust S1. Or it'll report S1 as a potential threat to the upper level.

In a multi-controller scenario, each controller only dominates their allocated switches. Therefore, a multi-controller can be cut(divided) into some independent single-controller scenario and the process still works in each sub-network.

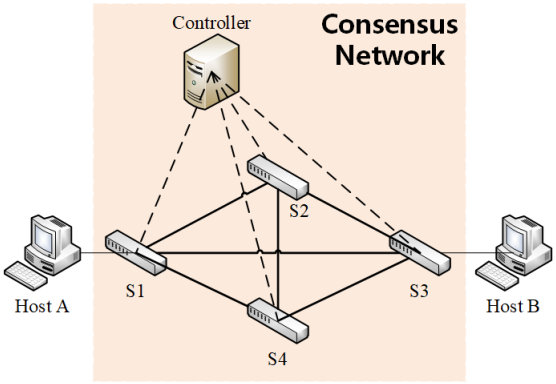


Fig. 3. Topology of Working Process

3.3 Statement of the Scheme

The network devices are mainly composed of controllers and switches. All devices in the network are blockchain nodes. Suppose that there are $c(c \geq 1)$ controllers and $d(d \geq 4)$ switches in the network, there will be $c + d$ nodes in total. In FRChain, the premise is that the content of the flow rule is correct. Therefore, we need to assume that the controller is safe, and other devices may have been compromised.

As for a data flow, the controller should generate $s(s \geq 1)$ flow entries to forward packets. We specify F as the processing strategy and f is a flow rule. Fand fshould have the following relation:

$$F = \{f_1, f_2, f_3, \ldots, f_s\} \tag{2}$$

Similarly, after finishing the strategy F, the controller sends F to each switching device and simultaneously synchronizes F into（to）blockchain network in the form of a transaction. The synchronized policy is H and we represent each flow rule as h. The correspondence is as follows:

$$H = \{h_1, h_2, h_3, \ldots, h_s\} \tag{3}$$

The key to FRChain is to compare F_n with H_n.

The attacker mainly attacks the traffic policy by adding, deleting, and modifying flow entries. Let's assumed that k flow rules in the policy F are attacked so that a malicious policy F' is made. The possible scenarios for F' are as follows:

(1) Delete k flow entries:

$$F' = \{f_1, f_2, f_3, \ldots, f_{s-k}\} \tag{4.1}$$

(2) Add k flow entries:

$$F' = \{f_1, f_2, f_3, \ldots, f_s, f_{s+1}, \ldots, f_{s+k}\} \tag{4.2}$$

(3) Modify k flow entries:

$$F' = \{f_1, f_2, f_3, \ldots, f_{s-k}, f'_{s-k+1}, \ldots, f'_s\} \tag{4.3}$$

In practical situations, the attack mode includes at least one of the above.

When $c = 1$, that is, in a single-controller network scenario, any data stream in the network should have F = H under normal network operation. Each block has a hash value of the asset. When device sw has $F_{sw} \neq H_{sw}$ or $Hash_{H_{sw}} \neq Hash(H_{sw})$, it'll trigger the vote request to the entire network and broadcasts a ticket. The data structure of the ticket is as follows:

$$Ticket_{sw} = \{ID_{sw}, H_{sw}, F_{sw}, T_1\} \tag{5}$$

ID_{sw}: the blockchain ID of device sw.

T_1: the timestamp.

After receiving the $Ticket_{sw}$and verifying the identity, all nodes except the controller compare H_{sw} and F_{sw} in the ticket with H_{local} that is stored in the local distributed database. The vote options are listed as follow:

Option A: $H_{sw} = H_{local}, F_{sw} \neq H_{local}$
Option B: $H_{sw} \neq H_{local}, F_{sw} \neq H_{local}$
Option C: $H_{sw} \neq H_{local}, F_{sw} = H_{local}$

After summarizing the voting results of nodes other than device sw, we can reach a conclusion:

(1) If the number of votes for A is more than half, it means that there might be a problem with F_{sw}. And the device sw may possibly be compromised. Besides, other nodes who do not vote for A may also be attacked.

(2) If more than half nodes vote for B, which is the worst situation by the way, it means H_{sw} and F_{sw} are both incorrect. The device is attacked, the flow policy has tampered, the distributed database is modified, and the blockchain network may also have security problems.

(3) When the number of vote C is more than a half, it means that F_{sw} has no problem, while H_{sw} has a problem. The distributed database is attacked, and the blockchain network may have also been attacked.

After reaching to the result, the controller will summarize the voting information and wait for the evidence which can prove F_{sw} is a benign flow entry.

The vote proposer sw will broadcast a signed ticket $Ticket_{sw\_signed}$ by using the administrator's private key Pri_key_{admin} to encrypt the hash result of $Ticket_{sw}$. The relation is shown as follow:

$$Ticket_{sw_{signed}} = Pri\_key_{sw}(Hash_{Ticket_{sw}}) \tag{6}$$

All nodes will verify the signature via the public key Pub_key_{sw}. The verification equation is shown below:

$$Hash_{Ticket'_{sw}} = Pub\_key_{sw}[Pri_{key_{sw}}(Hash_{Ticket_{sw}})] \tag{7}$$

If $Hash_{Ticket'_{sw}} = Hash_{sw}$ （$Ticket_{sw}$）, the device will pass the verification. If the proposer can't provide the $\mathrm{Hash}_{\mathrm{Ticket}_{\mathrm{sw}}}$, the result will be reported to administrators in form of Rpt_{sw}. The data structure is shown as follows:

$$Rpt_{sw} = \{T_{num}, s_{num}, o_{num}, a_{num}, ID_{sw}, H_{sw}, F_{sw}, T_1, T_2\} \tag{8}$$

- T_{num}: total number of nodes
- s_{num}: number of supportive votes
- o_{num}: number of opposing votes
- a_{num}: number of abandon votes
- T_2: a new timestamp.

So far, the administrators will get the result and take measures accordingly. Every time a verification procedure completes, the invalid flow entries in local database will be compressed as a series of hash value by using Merkle Tree algorithm to save space. The timeline of the scheme is shown in **Fig. 4**.

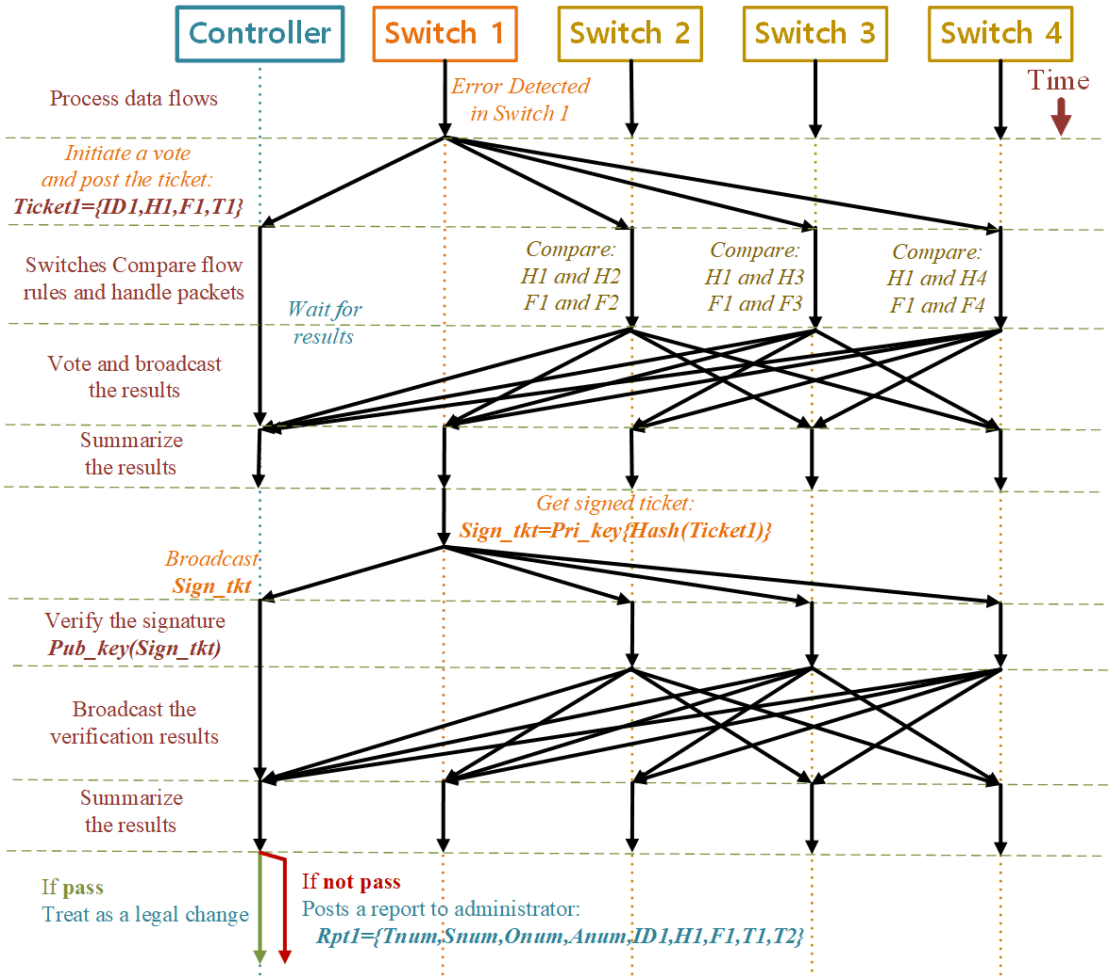


Fig. 4. The Timeline of the Process

When $c > 1$, which means it is in a multi-controller network, we consider using consortium blockchain in FRChain. No matter what kind of architecture of a multi-controller network is, a changeless principle of FRChain is to appoint the flow-entry maker as the transaction proposer. Now that we have presupposed that the prepared flow rules are correct, the procedure is similar to the single-controller scenario. Thus, we won't go into details here.

In summary, the scheme can directly detect whether the device sw has a problem, and can indirectly detect whether other devices have security threats. In this part, we detail the attacker's attack strategy and the defensive strategy adopted in the program. Next, we will

analyze why the blockchain is used and introduce some security features about FRChain.

3.4 Why Use Blockchain

In this subsection, we'll explain the security design and analyze some features of our scheme theoretically. In part 3.4.1, we'll illustrate how to create a new block by reaching the Tendermint consensus. Then how the consensus mechanism is used in FRChain and why we choose Tendermint as the consensus algorithm will be explained in part 3.4.2. Lastly, we'll depict why we choose distributed database instead of centralized ones to design FRChain.

3.4.1 How to Generate a New Block

As introduced above, the blockchain has been widely used in many aspects of new frontiers to deliver trusted service information in an untrusted environment. The first step to build a blockchain-based system is to reach a consensus. To avoid extra expenses, we use the Tendermint consensus algorithm. In FRChain, controllers and switches are all nodes. Additionally, we appoint the controller as an initiator to start the procedure. The specific process of generating a new block is illustrated in **Fig. 5**.

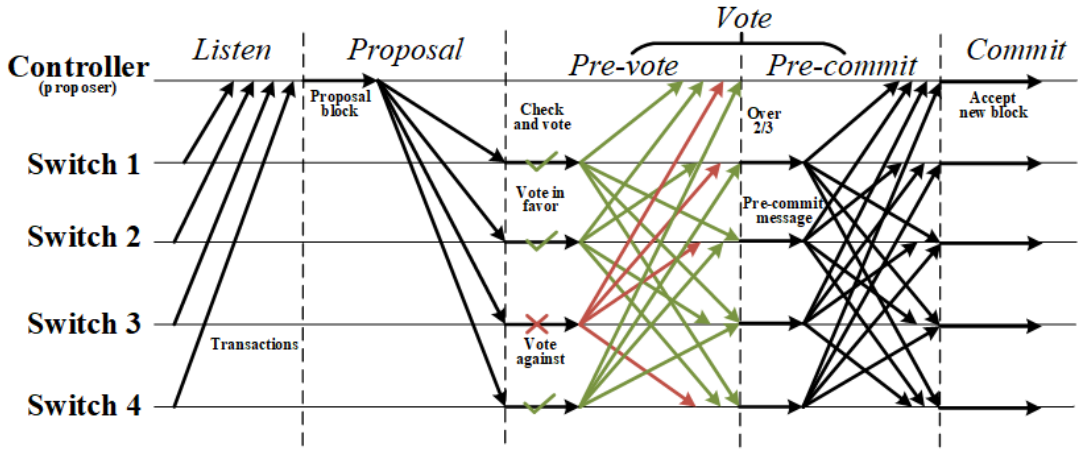


Fig. 5. Generate a New Block with the Tendermint Consensus Algorithm in FRChain

The Tendermint algorithm can statically set a set of validator nodes and then select one of them as the proposer node. The proposer node starts listening and collecting all transactions across the network. After a few minutes, assemble a new block and broadcast it to the entire network. This is the proposed block. After all the validator nodes of the whole network receive the proposal block, they start to read all the transactions in the block and verify them. If there is no problem, a pre-vote voting message is sent, indicating that the block is approved and a positive ticket will be voted. If there is an illegal transaction in the block, then a negative vote will be posted. These voting messages will be broadcast to all nodes, so each validator will issue a voting message and collect other's voting messages. If the percentage of vote in favor exceeds half of the valid votes, a pre-commit voting message will be sent. Now this is the second stage of voting. Each node is also listening and collecting pre-commit voting messages. When a validator node collects more than 1/2 of the pre-commit consent votes in valid votes, it means that the block is approved by most of the nodes. Then all participants can safely write this block to the local blockchain and append it to the end to complete the commit. It enters the next round and starts to propose a new block.

3.4.2 Consensus Mechanism in FRChain

During the process in FRChain, the usage of consensus can be divided into two stages.

1) FRChain appoints the controller as the only proposer to pack transactions and broadcast them to the entire network. As long as the proportion of malicious nodes in all valid nodes is smaller than 1/3, the message will be appended to the blockchain.
2) The switch can initiate a voting request to ask for a verification. Then other nodes except the controller participate in and broadcast the voting results to each other to determine whether the flow table is safe.

As introduced above, the Tendermint algorithm resembles the pBFT algorithm, therefore to some extent, both of them share similar features.

1) Firstly, the consensus mechanism is more applicable to the consortium chain or the private chain, because the number of nodes in the network is relatively fixed, and the node identity is comparatively determined. This feature is fairly suitable for an SDN-enabled network for it has stable devices and connections.
2) Secondly, in spite of the system's fault tolerance that can reach 1/3, it is still not high enough especially when compared with other consensus algorithms. This means once over 1/3 of nodes are compromised, the network will no longer be trustworthy and besides, we may not even know that the network is already in danger. Especially when some switches of the same type are deployed, similar configuration and vulnerability of these devices will increase the risk of multiple points failure.
3) Thirdly, as number of nodes in the network increases, system performance will drop rapidly because it takes longer to complete a vote. Therefore, our scheme is more suitable for smaller-scale networks.

3.4.3 Advantages of Using a Distributed Database

Distributed databases have already appeared before the emergence of blockchain. Decentralized system is a kind of distributed network architecture. Compared with a traditional database, a distributed database was designed to emphasize more on security, rather than data management function, like data retrieval.

Traditional database always serves as an independent device, which means, once the centralized database being attacked, all data will be posed into threats. Along with high performance in data processing, blockchain can issue consensus in untrustworthy environments rapidly and stably. In contrast with the centralized solutions like setting the controller as the trust root, the decentralized scheme might have lower performance but can be more effective in defending the links attacks, like the MitM attack. In FRChain, we use BigchainDB because it can be deployed in a small-scale network so that the synchronization process will be much faster.

But distributed databases have their shortage. Implementing a consensus mechanism needs to take up a certain storage space and sacrifice some performance of the system. Though Merkle tree [31] was used in blockchain to save disk space, there still not enough place for network devices in the long run. However, if the verified data, like flow rules, is compressed in the form of a Merkle tree at regular time intervals, then enough space can be freed up to store new content.

4. Evaluation Results and Analysis

This section depicts the evaluation and performance of FRChain. In subsection 4.1, we'll introduce the simulation environment, conduct the experiment and show how to initiate the system. After that, the performance evaluation including time cost of system, communication latency and hardware overhead will be illustrated in subsection 4.2. At last in subsection 4.3, a performance conclusion of FRChain and its practical scenario will be issued.

4.1 Simulation Environment and Initiation

We built the system on five virtual machines with ubuntu 16.04LTS as operating system. One of them was set up as the RYU controller and the other four were implemented as Open vSwitches. The device specifications are listed in **Table 1**.

Table 1. The Device Specifications

Devices	Specifications	
Ryu	Amounts	1
	CPU	Intel i7-8850H
	Core	2
	RAM	1G
	Version	4.30
Open vSwitch	Amounts	4
	CPU	Intel i3-3110M
	Core	1
	RAM	1G
	Version	2.8.1

A proper initialization is important to FRChain because we need to make sure all nodes start the system at the same height of blockchain. Details are explained as follows:

1) Set up the environment of SDN and BigchainDB, initialize the whole network and tested the connectivity.
2) Optimize the source code of the Ryu controller and the Open vSwitch and then integrate BigchainDB into the system.
3) Edit the genesis block to ensure all following blocks could be linked into an expected place.
4) Network administrators authenticate other nodes and assigned various rights to controllers and other SDN devices.
5) The authenticated nodes are raised as validators to reach consensus with the same genesis file.
6) A number of blocks should be generated first before the system is put into use.

Now the controller can post transactions that contain key information of flow entries and broadcasts them to the BigchainDB network. Then original flow rules will be kept in the decentralized database for other nodes to track.

4.2 Performance Evaluation

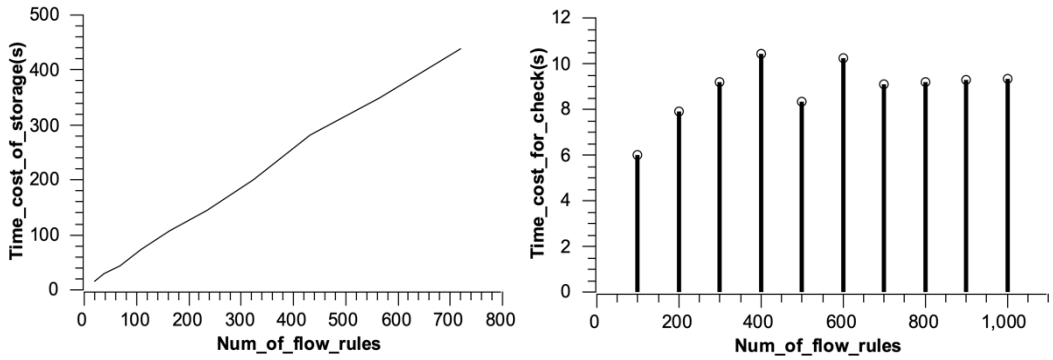
In this subsection, we will illustrate the performance simulation and its evaluation. Firstly, we'll represent the time cost for FRChain of dealing with different number of flow rules. Secondly, the influence of network performance caused by FRChain will be shown in both controller side and the switch side. Thirdly, we'll evaluate the overhead of hardware and analysis the trends respectively.

4.2.1 Time Cost of Processing Flow Entries

In this part, we tested the time cost of storage and verification. We use the number of flow rules as the independent variable and the reaction time as the dependent variable to figure out how the time cost changes with the increase of flow entries. The experimental results are shown in **Fig. 6**.

To test the performance, we develop a traffic generator in Python, which will continuously generate packets with different destination addresses, so that the switches will thereby produce packet-in messages to query for instructions. Then the controller will have to generate new flow rules rapidly.

In **Fig. 6(a)**, the time cost by storing flow rules increases approximately linearly as the number of flow rules increases because the number of processing flow rules per unit time is almost equal. **Fig. 6(b)** shows the time cost by performing a verification with different amounts of flow entries. It can be noticed that when the number of flow rules is less than 300, the time required to perform verification increases. However, after exceeding 300, the time cost is basically maintained at about 10s.



(a) Time cost of storing flow rules (b) Time cost of checking flow rules

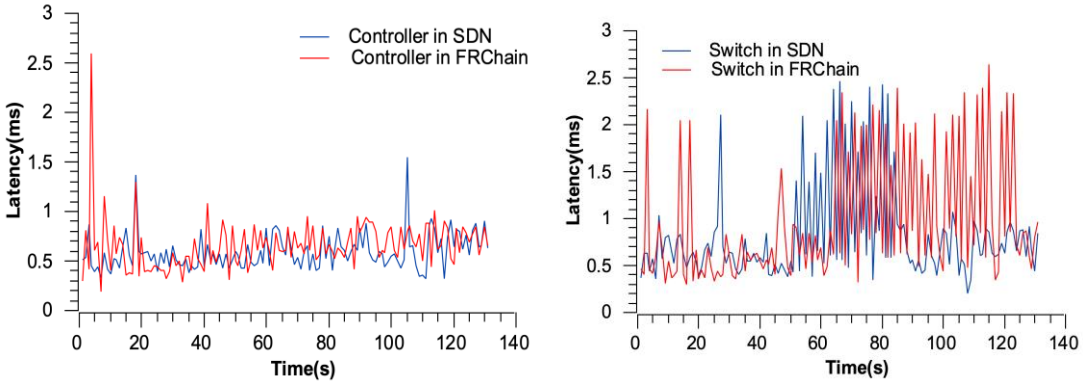
Fig. 6. Time Cost of Storing and Checking Flow Rules

4.2.2 Latency Contrast between SDN and FRChain.

We tested the latency in both native SDN environment and FRChain. The results are listed below in **Fig. 7**. It can be concluded that the average delay on the controller side is lower than that on the switch side.

At the beginning of **Fig. 7(a)**, a pulse appears and that's because a large amount of flow rules will be created during the initialization, many transactions need to be posted by one node at the same time, which might cause temporary congestion. After a short time, the latency had returned to a normal level.

In **Fig. 7(b)**, The pulse at the beginning also happened in this way. In the middle and rear part, it can be seen that the network average latency has improved fiercely and the increased delay in FRChain is similar to that in the traditional SDN. But the duration of the pulses in FRChain is longer than that in SDN. That is because we probed the situation of connections between devices so that some new flow rules were delivered separately. Increased delay can ensure that flow rules are written into the blockchain and achieve network-wide synchronization.



(a) Latency of controller (b) Latency of switch

Fig. 7. The Contrast of Latency between Controller and Switch in SDN and Our Scheme

4.2.3 Overhead Evaluation and Analysis

We tested the overhead of FRChain including the usage of CPU and memory. The results are shown in **Fig. 8**. Since our scheme works in software-based SDN devices, the capability varies depends on the hardware performance.

We have conducted multiple experiments with handling different numbers of flow entries. The experimental results show that when the number of flow rules is small, the CPU overhead caused by the execution of the scheme will increase as the number of flow rules increases. However, when the number of flow rules reaches 39, the CPU usage of the solution tends to be stable and the value remains at about 5.75%.

The situation in memory usage is similar to CPU. The difference lies in that regardless of whether the flow rule is processed, this solution will cause about 7.73% memory overhead to keep main thread working. The memory usage will increase with the increasing number of flow rules. But the slope of the growth curve is small, which means when processing a large number of flow rules, like over 2000, the memory usage will not affect the normal operation of the system.

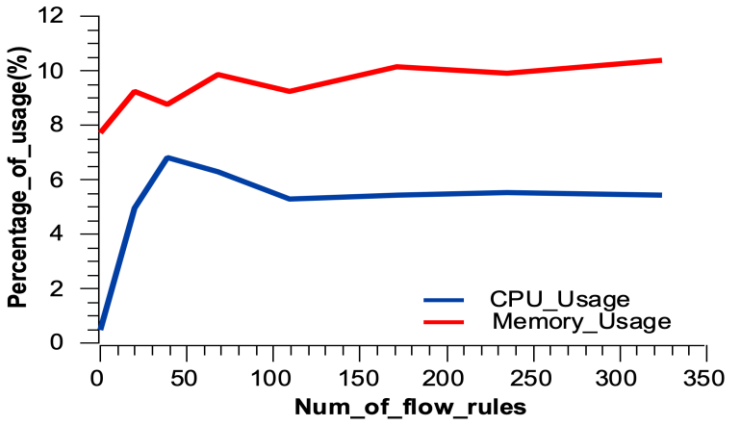


Fig. 8. Usage of CPU and Memory

4.3 Evaluation Results

Due to the lack of relevant solutions in solving data forwarding security problems in a similar way, we are not able to compare FRChain with other schemes. Thus, we tested related indicators of performance of our scheme and made a comparison with overhead of Ryu and Open vSwitch core process. The results are shown in **Table 2**. The "Min" represents the guardian process of the applications while "Stable" indicates approximately stable value of each item when they are busy. And the missing values means they varies irregularly. It can be concluded that both Ryu and Open vSwitch need more CPU consumption to finish complex calculation like planning path for data flows in controller and loading new strategies in switch. As for the FRChain, it only needs more buffer to produce transactions.

Table 2. Comparation of Hardware Overhead

Process	Status	CPU usage	RAM usage
Ryu	Min	0.3%	4%
	Stable	-	4.3%
Open vSwitch	Min	0.3%	0.3%
	Stable	-	4.4%
FRChain	Min	0.5%	7.8%
	Stable	5.5%	-

Generally, FRChain can cause some hardware overhead while ensuring the normal operation of the system. In an SDN network, the flow rules are not generated uniformly over time. Therefore, in the practical environment, a large number of flow entries will be generated during a short period, which will cause additional overhead for the network and hardware. However, after finishing the initiation stage, the system steps into a stable phase and the existing flow tables in data plane is temporarily sufficient for processing most of flows, especially in a small-scale network. Therefore, the latency and hardware overhead will be relatively stable in the late stage and will not affect the performance of the SDN.

DistBlockNet [23] is a similar solution to FRChain. DistBlockNet saves the newest flow rules issued by the controller to the blockchain for reference, and its main work focuses on the deployment of multiple security strategies. As for the performance test, the peak CPU usage of DistBlockNet is more than 13%, and the average value is over 8%. In the case that the hardware configuration used by DistBlockNet is not yet clear, all devices in our test are deployed with a single-core CPU and 1G memory. The test results show that the peak CPU usage rate is less than 8%, and the average value is about 6%.

Another similar solution is the controller distributed audit system that was proposed by Guan Z et al. in [32]. In that scheme, the blockchain is mainly used to store the flow rules and logs issued by the controller to enhance the audit function. But no function and performance test are mentioned in the paper.

Application scenario. FRChain is suitable for small-scale SDN networks, for example, to protect the security of data transforming process between enterprises. As for a large cross-regions network, a single controller is usually not capable enough for traffic scheduling. Instead, the multi-controller clusters might be a suitable choice. Under this circumstance, FRChain is also applicable. By considering the different regions where diverse controllers are obligated to as a small single-controller subnet. And different subnets have their own chains respectively. Then FRChain will be able to provide protection for different subnets.

5. Conclusion and Future Work

After studying the relevant research, combined with the advantages and disadvantages of various schemes, we believe that the key to solving the data forwarding security problem in SDN is how to keep flow rules safe. Based on the characteristics of blockchain and existing application scenarios, we propose FRChain to keep flow strategies safe by storing key information of original flow rules into blockchain in the form of transactions. During the process, we use the consortium chain and private chain to maintain the central status of the SDN controllers and transform the contradiction between the centralized architecture and the decentralized architecture. We simulated the proposed scheme by optimizing the source code of RYU controller, Open vSwitch and BigchainDB. Under the premise of ensuring the normal operation of the security function, Although the system will cause additional overhead, FRChain doesn't affect the normal operation of the SDN and keep the consumption of both communication and hardware resources within a relatively normal range.

In the future, we will take measures to design a more efficient way to deal with the conflict of benign flow rules and try to make FRChain more applicable to new protocols. Then improve the pre-warning capability of the system and try to improve tracking capabilities by collecting the attacker's relevant information as detailed as possible. Another move is to improve the compatibility between software and reduces the consumption of system resources. Besides, we will choose some other blockchains and distributed database to compare with BigchainDB in order to improve the performance.

References

[1] M. Dhawan, R. Poddar, K. Mahajan, and V. Mann, "SPHINX: Detecting Security Attacks in Software-Defined Networks," in *Proc. of The Network and Distributed System Security Symposium (NDSS)*, vol. 15, pp. 8-11, Jan. 2015. Article (CrossRef Link)

[2] S. Shin, V. Yegneswaran, P. Porras, and G. Gu, "Avant-guard: Scalable and vigilant switch flow management in software-defined networks," in *Proc. of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, pp. 413-424, Nov. 2013. Article (CrossRef Link)

[3] S. Hong, L. Xu, H. Wang, and G. Gu, "Poisoning Network Visibility in Software-Defined Networks: New Attacks and Countermeasures," in *Proc. of The Network and Distributed System Security Symposium (NDSS)*, pp. 8-11, Feb. 2015. Article (CrossRef Link)

[4] N. Mckeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69-74, Apr. 2008. Article (CrossRef Link)

[5] T. Zhang, A. Bianco, P. Giaccone, and A. P. Nezhad, "Dealing with misbehaving controllers in SDN networks," in *Proc. of GLOBECOM 2017-2017 IEEE Global Communications Conference*, pp. 1-6, Dec. 2017. Article (CrossRef Link)

[6] A. Ranjbar, M. Komu, P. Salmela, and T. Aura, "An SDN-based approach to enhance the end-to-end security: SSL/TLS case study," in *Proc. of NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium*, pp. 281-288, July 2016. Article (CrossRef Link)

[7] M. Antikainen, T. Aura, and M. Sarela, "Spook in your network: Attacking an SDN with a compromised openflow switch," in *Proc. of Nordic Conference on Secure IT Systems*, vol. 8788, pp. 229-244, 2014. Article (CrossRef Link)

[8] Y. Wang, J. Bi, and K. Zhang, "A tool for tracing network data plane via SDN/OpenFlow," *Science China (Information Sciences)*, vol. 60, no. 2, pp. 74-86, Feb. 2017. Article (CrossRef Link)

[9] M. Wang, J. Liu, J. Mao, H. Cheng, J. Chen, and C. Qi, "Route Guardian: Constructing Secure Routing Paths in Software-Defined Networking," *Tsinghua Science and Technology*, vol. 22, no. 4, pp. 400-412, Aug. 2017. Article (CrossRef Link)

[10] T. Wang and H. Chen, "SGuard: A Lightweight SDN Safe-Guard rchitecture for DoS Attack," *China Communications*, vol. 14, no. 6, pp. 113-125, June 2017. Article (CrossRef Link)
[11] E. Alshaer and S. Alhaj, "FlowChecker: Configuration analysis and verification of federated OpenFlow infrastructures," in *Proc. of the 3rd ACM Workshop on Assurable and Usable Security Configuration*, pp. 37-44, Oct. 2010. Article (CrossRef Link)
[12] S. Son, S. Shin, V. Yegneswaran, P. Porras, and G. Gu, "Model checking invariant security properties in OpenFlow," in *Proc. of IEEE International Conference on Communications*, pp. 1974-1979, Nov. 2013. Article (CrossRef Link)
[13] A. Khurshid, W. Zhou, M. Caesar, and P. B. Godfrey, "Veriflow: Verifying network-wide invariants in real time," in *Proc. of the 1st Workshop on Hot Topics in Software Defined Network*, pp. 49-54, 2013. Article (CrossRef Link)
[14] P. W. Chi, C. Kuo, J. W. Guo, and C. L. Lei, "How to detect a compromised SDN switch," in *Proc. of 2015 1st IEEE Conference on Network Softwarization (NetSoft)*, pp. 1-6, 2015. Article (CrossRef Link)
[15] Z. Zhibin, C. Chaowen, and Z. Xianwei, "A Software-Defined Networking Packet Forwarding Verification Mechanism Based on Programmable Data Plane," *Journal of Electronics & Information Technology*, vol. 42, no. 5, pp. 1110-1117, 2020. Article (CrossRef Link)
[16] X. Qiu, K. Zhang, and Q. Ren, "Global Flow Table: A convincing mechanism for security operations in SDN," *Computer Networks*, vol. 120, pp. 56-70, 2017. Article (CrossRef Link)
[17] T. Sasaki, C. Pappas, T. Lee, T. Hoefler, and A. Perrig, "SDNsec: Forwarding Accountability for the SDN Data Plane," in *Proc. of 2016 25th International Conference on Computer Communication and Networks (ICCCN)*, pp. 1-10, Sep. 2016. Article (CrossRef Link)
[18] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *White Paper Bitcoin*, pp. 1-9, 2009. Article (CrossRef Link)
[19] K. Kataoka, S. Gangwar, and P. Podili, "Trust list: Internet-wide and distributed IoT traffic management using blockchain and SDN," in *Proc. of 2018 IEEE 4th World Forum on Internet of Things (WF-IoT)*, pp. 296-301, May 2018. Article (CrossRef Link)
[20] C. Tselios, I. Politis, and S. Kotsopoulos, "Enhancing SDN security for IoT-related deployments through blockchain," in *Proc. of 2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, pp. 303-308, Nov. 2017. Article (CrossRef Link)
[21] M. Ali, J. Nelson, R. Shea, and M. J. Freedman, "Blockstack: A global naming and storage system secured by blockchains," in *Proc. of 2016 USENIX Annual Technical Conference*, pp. 181-194, June 2016. Aritcle (CrossRef Link)
[22] P. K. Sharma, M. Y. Chen, and J. H. Park, "A software defined fog node based. distributed blockchain cloud architecture for IoT," *IEEE Access*, vol. 6, pp. 115-124, Sep. 2017. Article (CrossRef Link)
[23] P. K. Sharma, S. Singh, Y. Jeong, and J. H. Park, "Distblocknet: A distributed blockchains-based secure SDN architecture for IoT networks," *IEEE Communications Magazine*, vol. 55, no. 9, pp. 78-85, Sep. 2017. Article (CrossRef Link)
[24] J. Chen, "Flowchain: A distributed ledger designed for peer-to-peer IoT networks and real-time data transactions," in *Proc. of the 2nd International Workshop on Linked Data and Distributed Ledgers (LDDL2)*, pp. 1-10, Jan. 2017.
[25] M. Singh and S. Kim, "Blockchain based intelligent vehicle data sharing framework," *arXiv: Cryptography and Security*, July 2017. Article (CrossRef Link)
[26] A. Yazdinejad, R. M. Parizi, A. Dehghantanha, and K. R. Choo, "Blockchain-enabled Authentication Handover with Efficient Privacy Protection in SDN-based 5G Networks," *IEEE Transactions on Network Science and Engineering*, p. 1, Aug. 2019. Article (CrossRef Link)
[27] Z. A. El Houda, A. S. Hafid, and L. Khoukhi, "Cochain-SC: An Intra- and Inter-Domain Ddos Mitigation Scheme Based on Blockchain Using SDN and Smart Contract," *IEEE Access*, vol. 7, pp. 98893-98907, July 2019. Article (CrossRef Link)

[28] X. Zhang and X. Chen, "Data Security Sharing and Storage Based on a Consortium Blockchain in a Vehicular Ad-hoc Network," *IEEE Access*, vol. 7, pp. 58241-58254, Jan. 2019. Article (CrossRef Link)
[29] J. Kang, R. Yu, X. Huang, M. Wu, S. Maharjan, S. Xie, and Y. Zhang, "Blockchain for Secure and Efficient Data Sharing in Vehicular Edge Computing and Networks," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4660-4670, June 2019. Article (CrossRef Link)
[30] T. McConaghy, R. Marques, A. Müller, D. D. Jonghe, T. McConaghy, G. McMullen, R. Henderson, S. Bellemare, and A. Granzotto, "BigchainDB: a scalable blockchain database," *White Paper, BigChainDB*, 2016. Article (CrossRef Link)
[31] M. Szydlo, "Merkle tree traversal in log space and time," in *Proc. of International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 541-554, 2004. Article (CrossRef Link)
[32] Z. Guan, H. Lyu, H. Zheng, D. Li, and J. Liu, "Distributed Audit System of SDN Controller Based on Blockchain," in *Proc. of International Conference on Smart Blockchain*, vol. 11911, pp. 21-31, 2019. Article (CrossRef Link)

Weichen Lian received his B.S. degree from the Department of Electronics and Communication Engineering, Beijing Electronics Science & Technology Institute in 2016. He is currently an M.S. candidate from Beijing Electronics Science & Technology Institute. His research interests include network security, software-defined networking and blockchain.

Zhaobin Li received his B.S. degree from Beijing Electronics Science & Technology Institute, in 2001, the M.S. degree from Beijing University of Posts and Telecommunications, in 2006, and the Ph.D. degree from Beijing University of Posts and Telecommunications, in 2009. He is currently an associate professor with the Department of Communication Engineering, Beijing Electronics Science & technology Institute, China. He has published papers in Acta Electronica Sinica, Journal on Communications, Journal of Electronics & Information Technology, Application Research of Computers, Computer Engineering. His research interests include software-defined networking, network security, blockchain technology, cloud computing, and Internet exchange. He is a reviewer for 6 journals and 10 conferences.

Chao Guo received her B.S. degree from the North China Institute of Science and Technology in 2009. She obtained a Ph.D. degree in the Department of Communication Engineering, School of Computer & Communication Engineering, from the University of Science and Technology Beijing in 2015. Now, she is a lecturer in the Department of Electronics and Communication Engineering, Beijing Electronics Science and Technology Institute, P.R. China. Her research interests include transmission control scheme, congestion control, and communications security.

Zhanzhen Wei born in 1971. He is currently a professor with the Department of Communication Engineering, Beijing Electronics Science & technology Institute, China. He has participated in many national information security issues and served as the team leader, and has won many provincial and ministerial second and third prizes. His research interests include network security, trusted network, and software-defined networking.

Xingyuan Peng received her B.S. degree from the Department of Communication Engineering, Shanghai University in 2018. She is currently an M.S. candidate from Beijing Electronics Science & Technology Institute. Her research interests include network security, software-defined networking,and blockchain.