Computer Science and Information Systems 19(2):523–545 https://doi.org/10.2298/CSIS210222001G

A novel Security Mechanism for Software Defined Network Based on Blockchain

Xian Guo, Chen Wang, Laicheng Cao, Yongbo Jiang, and Yan Yan

School of Computer and Communication,
Lanzhou University of Technology, Gansu 730050, China
iamxg@163.com, {572957016, 28140795, 670342320, 414696390}@qq.com,

Abstract. The decoupling of the data plane and the control plane in the Software-Defined Network (SDN) can increase the flexibility of network management and operation. And it can reduce the network limitations caused by the hardware. However, the centralized scheme in SDN also can introduce some other security issues such as the single point of failure, the data consistency in multiple-controller environment and the spoofing attack initiated by a malicious device in the data plane. To solve these problems, a security framework for SDN based on Blockchain (BCSDN) is proposed in this paper. BCSDN adopts a physically distributed and logically centralized multi-controller architecture. LLDP protocol is periodically used to obtain the link state information of the network, and a Merkle tree is establised according to the collected link information and the signature is generate based on KSI for each link that submitted by a switch by the main controller selected by using the PoW mechanism. Such, the dynamic change of network topology is recorded on Blockchian and the consistency of the topology information among multiple controllers can be guaranteed. The main controller issues the signature to the corresponding switch and a controller checks the legitimate of a switch by verifying the signature when it requests the flow rule table from the controller later. The signature verification ensures the authenticated communication between a controller and a switch. Finally, the simulation of the new scheme is implemented in Mininet platform that is a network emulation platform and experiments are done to verify our novel solution in our simulation tool. And we also informally analysis the security attributes that provided by our BCSDN.

Keywords: SDN, LLDP, Blockchain, KSI.

1. Introduction

With the rapid development of Internet application, the Software-Defined Networking (SDN) is coming into being to meet the increasing demand for the network traffics and have been widely applied in many areas [1-5]. While the SDN provides convenience in network management and operation, the split of the control plane and the data plane also maybe result in some security issues [3, 6-9], such as the single points of failure in the control plane, and the fragile channels between the control plane and the data plane, the data consistency in multiple controllers environment and all kinds of attacks initiated by a malicious switch and so on. The control plane is the most important component of

SDN. It is a key research area to resolve the problems that mentioned above. The redundant manner that uses the multi-controller architecture is a generic solution, such as the master-slave backup [10-13] and Byzantine Fault Tolerance scheme (BFT) [14-17]. In the master-slave backup architecture, these two controllers that are respectively called the master controller and the slave controller are used. When the master controller is shut down because of some failures or other reason, the slave controller is initiated and replaces the master controller to provide the network management task. The mechanism can ensure the data consistency and can improve the resiliency of the control layer. However, the method can't fundamentally solve the single point of failure. In [14-17], Byzantine fault-tolerant mechanisms based on Byzantine protocol are proposed to achieve the data consistency on different controllers and to confirm a failure controller. However, in these mechanisms, when the main controller fails, a view shift needs to be performed, which can cause a great overhead of network resources.

Blockchain [18-19] is a proof-tamper, and distributed database that is jointly maintained by multiple parties. It can achieve credible data sharing without the participation of a Trusted Third Party (TTP) and can increase the scalability and flexibility of the network. A survey about deployment of Blockchain in SDN is done in [20]. To solve the security problem of the control plane in SDN, some solutions based on Blockchain are proposed in [21-29]. However, these schemes still adopts the native structure of SDN about the internal structure of the controller, so they can't fully take advantage of the Blockchain features, and don't provide a systematic security mechanism.

In addition, the Link Layer Discovery Protocol (LLDP) [30] is a standard protocol of the network topology. However, some attacks such as the switch spoof, LLDP flood and so on, are found in [31]. Secondly, the establishment of a secure channel between the control plane and the data plane in SDN is also a hot research direction. The Transport Layer Security Protocol (TLS) [32] used by the Openflow protocol is by default a protocol between a controller and a switch. However, due to the complex configuration and the communication cost, TLS is considered as an alternative solution in later versions, which often lead to some security issues such as DDoS attack [33].

Keyless Signatures Infrastructure (KSI) [34] is a globally distributed system for providing timestamp and server-supported digital signature service. Only hash operation is used in KSI, so the scheme will not be impacted by some security problems such as a key leakage. KSI can ensure the long-term validity of digital signature and often is used for achieving a reliable communication. Using of KSI can prevent some attacks such as the switch spoof and so on. However, although KSI provides a complete cryptographic system, the core layer must calculate the root of the hash tree generated every time and publish it in the database. At present, there are still problems such as the lack of a credible mechanism for the database, the release cycle cannot meet the more fine-grained requirement, and the release channel must be a secure channel. The combination of Blockchain and KSI can solve these above problems. The hash calendar (Merkle tree) generated when the network topology change can be stored on Blockchain.

Aiming to the SDN security issues discussed before, a security framework for SDN based on Blockchain (BCSDN) is proposed in this paper. BCSDN adopts a physically distributed and logically centralized multi-controller architecture, and uses blockchain technology to build a unified database among controllers. All nodes in the network will periodically collect the link state information according to the instruction comes from the main controller selected by the consensus scheme PoW [35-36]. A Merkle tree will be set up according to the link state information and the signature will be generated for each switch by the main controller based on KSI. The data consistency among multiple controllers can be ensured by using of the Blockchain. And the root hash value generated in every round will be written into the block header. The main controller will issue the signature to the corresponding switch. The proof-tamper, auditable and traceable features of Blockchain and KSI's features provide security guarantees. Finally, the simulation of BCSDN is implemented in Mininet platform [37] that is a network emulation platform and experiments are done to verify our novel solution in our simulation tool. And we informally analysis the security attributes that provided by our BCSDN.

The rest of the paper is organized as follows. Section 2 presents the related work. Some backgrounds on our solution are given in section 3, followed by our new scheme BCSDN in section 4. In section 5, security attributions of BCSDN are informally analyzed. Implementation and performance analysis of our solution are introduced in section 6. Finally we conclude our work in section 7.

2. Related Work

A redundant manner that uses the multi-controller architecture is a generic solution to resolve the single point of failure in SDN. The architecture is a scalable control plane solution for the large-scale SDN. To achieve high resilience, an SDN switch can connect one master controller for normal operation and one slave controller that backup the function of the master controller. Once the master controller fails, one of the slave controllers will be assigned to switches to works as the new master controller. However, the inappropriate slave controller assignment may cause controller chain failure, where running out of the capacity of the assigned controller, even crash the entire network. In [10], a dynamic slave controller assignment that prevents the network crash by planning slave controller assignment ahead of the controller failures is proposed. The controller chain failure phenomenon that incurred by unreasonable slave controller assignment can be solved. The slave controller assignment problem is formulated as a multi-objective mixed optimization problem that considers multiple network factors such as latency, load balancing and robustness. And it has been proven that it is a NP-complete complexity problem. A dynamic slave controller assignment (DSCA) scheme is introduced in [10]. DSCA firstly checks whether there are controller failures in state detection module, then completes the elastic slave assignment and generates a new slave assignment for switches in efficient slave assignment module. Finally, in role adjustment module, it changes the roles of some controllers and reconnects switches. Simulation results show DSCA can decrease the worst case latency under controller failures by 35.1% averagely, and reduce the probability of network crash.

In multi-controller architecture, the uneven distribution of traffic load in the controllers can degrade system performance. In [11], a self-adaptive load balancing (SALB) scheme that balances load among multiple controllers dynamically with multiple switch migration from source controllers to target controllers is proposed. The key feature of SALB is an effective distribution of load under high load condition while considering the distance between switches and target controllers simultaneously. The efficacy of SALB is demonstrated through experimentation in [11] and the experimental results show that SALB experiences a small number of packet drops, which is less than 1.23% of the total number of message exchanges among the controllers.

Robustness and fault tolerance are two important metrics to be considered in assessing SDN’s advantage. The currently available SDN controllers offer different fault tolerance mechanisms. In [12], existing fault-tolerant SDN controller solutions are surveyed and a mechanism is proposed to design a consistent and fault-tolerant Master-Slave SDN controller. The scheme [12] is able to balance consistency and performance. The main objective of [12] is to bring the performance of an SDN Master-Slave controller as close as possible to the one offered by a single controller. This is achieved by introducing a simple replication scheme, combined with a consistency check and a correction mechanism, that influence the performance only during the few intervals when it is needed, instead of being active during the entire operation time.

Despite many advantages of SDN, its deployment in the practical field is restricted since reliability and fault-tolerance capabilities of the system are not satisfactory. To overcome these difficulties of SDN, an architecture called FT-SDN has been proposed in [13]. The proposed architecture consists of a simple and effective distributed Control Plane with multiple controllers. FT-SDN uses a synchronized mechanism to periodically update the controller’s state within themselves. In case of failure, FT-SDN has the ability to select another working controller based on the distance and delays among different network entities.

In the multi-controller architecture, most of state synchronization processes on different controllers depend on the assumption of a correct decision-making in the controllers. Successful introduction of SDN in the critical infrastructure networks also requires catering to the issue of unavailable, unreliable (e.g. buggy), and malicious controller failures. A framework tolerant to unavailability and Byzantine failures is proposed in [14]. It is called as MORPH. The MORPH can distinguish and localize faulty controller instances and appropriately reconfigure the control plane. A prototype SDN controller that can tolerate Byzantine faults in both the control and data planes is proposed in [15]. The performance of the novel solution is compared with current standard fault vulnerable open source SDN controllers. The experiment shows there is a reasonable slowdown of [15] as is expected in the transition from a fault vulnerable to a fault tolerant design. Their best controller can show only a 2x slowdown even though it only need 4 replica components, and so it can tolerate a single compromised component without affecting control and/or forwarding decisions in the networks. However, controllers in [15] are not fit for high performance levels to be adopted in large-scale networks.

A security framework based on the Byzantine protocol is proposed in [16]. In the scheme, controllers execute the Byzantine protocol and each switching device is managed by a controller view. The control information is given after multiple controllers arbitrate. By quantifying the heterogeneity of controllers, a two-stage controller view election algorithm is designed to ensure the availability of the network and the security of views.

Network survivability is the ability to maintain service continuity in the presence of failures. In [17], the network survivability of SDN is discussed in disaster situations. The solution in [17] considers multi-controller failure and the mechanism can reduce the non-operational network devices in disaster situations. Preliminary results show that, by applying the proposed new approach, it is possible to achieve substantial improvements in network survivability.

To resolve security and privacy issues in SDN, some solutions based on Blockchain have explored. In [21], Blockchain Security over SDN (BSS) is proposed which protects privacy and availability of resources against non-trusting members. To verify their solution, mininet emulator is used for simulating custom SDN network topology. OpenDaylight controller is integrated with OpenStack controller. For testing purpose of Blockchain, Pyethereum tester tool under Ethereum platform is implemented. Serpent programming is used for creating contract in the blockchain. The simulation result shows that BSS facilitates files sharing among SDN users in distributed peer-to-peer basis using OpenStack as a cloud storage platform.

Since the large number of devices connected to the Internet of things (IoT) networks, the SDN-based network architecture makes the deployment and configuration of IoT much easier. In the IoT network, the fine-grained network traffic is critical to network management. In [22], a novel scheme based on Blockchain is proposed to measure the fine-grained network traffic in the SDN-based IoT networks and to ensure the security and consistency of the statistics. To measure flow traffic with low overhead and high accuracy, an ARIMA model and forecast the network traffic with the coarse-grained measurement of flows is designed. An objective function in ARIMA mode can decrease the estimation errors. A heuristic algorithm to obtain the optimal solution of the fine-grained measurement is used due to the objective function is an NP-hard problem.

To improve forwarding efficient of devices in the data plane of SDN, a method called TrustBlock is proposed in [23], which introduces trust as a security attribute in SDN routing planning. Besides, in order to enhance the integrity and controllability of trust evaluation, the double-layer blockchain architecture is established in [23]. In the first layer, the behavior data of the node is recorded, and then the trust calculation is performed in the second layer. In the evaluation model, nodes’ trust is calculated from three aspects: direct trust, indirect trust and historical trust. Firstly, from the perspective of security, blockchain is used to achieve identity authentication of nodes, after that, from the perspective of reliability, the forwarding status is used to calculate the trust value. Secondly, consensus algorithm is used to filter malicious recommendation trust value and prevent colluding attacks. Finally, the adaptive historical trust weight is designed to prevent the periodic attack. In [23], the entropy method is used to determine the weight of each evaluation attribute, which can avoid the problem that the subjective judgment method is not adaptable to the weight setting. Simulation results show that the detection rate of the TrustBlock is up to 98.89%, which means this model can effectively identify the abnormal nodes in SDN. Moreover, it is attractive in terms of integrity and controllability.

In Software-Defined Networking (SDN), Northbound Interface provides APIs, which allow network applications to communicate with SDN controllers. However, a malicious application can access to SDN controller and perform illegal activities via these APIs. Although some studies proposed AAA (Authentication, Authorization, Accounting) systems to protect SDN controllers from malicious applications, their proposed systems also exist several limitations. Attackers can compromise a system, then modify its database or files to gain higher privileges. This system can be taken down because of Single Point of Failure threat. A novel system BlockAS is proposed to improve security for the Northbound interface in [24]. It is used to authenticate, authorize and monitor accessing critical controller resources from applications. Specifically, BlockAS leverages Blockchain features to maintain the immutability and decentralization of credential data. In SDN, the lack of consistent records of network data poses difficulties for network management, and heterogeneous device heterogeneity poses a hindrance to software-defined network interoperability. [25] summarizes the development status and existing problems of software-defined network, proposes, realizes distributed consistent record of software-defined network data, and breaks the multi-vendor device isolation for fault recovery. Reduce the cost of network failure recovery and achieve unified scheduling of business capabilities. A security framework is also proposed that integrates Blockchain technology with multi-controller SDN in [26]. The main idea of the framework is to associate a set of controllers to each domain and to ensure a secure and trustworthy inter-controller communication. So, the proposed architecture considers a master controller and redundant controllers for each network domain. The architecture also integrates a reputation mechanism to identify a malicious controller. In [27], a distributed Blockchain-based SDN-IoT enabled architecture is proposed. It is the main goal of this framework to manage smart building. The traditional approach that manages the health-related data is often the centralized approach. It is not convenient to share and process electronic health data across the different institutions. In [28], an alternative way based on Blockchian technology is proposed to deal with information exchange across multiple stakeholders. A Blockchain-enabled Packet Parser (BPP) of the SDN is proposed in [29]. The scheme not only can detect attack in SDN and also can implement Blockchain protocol in data plane.

3. Research Background

3.1. The SDN Architecture

To resolve some issues in traditional network architecture, Software Defined Networks (SDN) is proposed. SDN is an emerging network architecture that decouples the control plane from the data plane and provides a software-based centralized controller. By this separation of control plane and data plane, switches in network become simple forwarding devices. Whereas, routing decision making is shifted to the controller, which can provide a global view of the network and a programming abstraction. This centralized entity provides a capability that an operator can program and real-time control underlying networks and devices. By using SDN, the network management becomes simply and helps in removing rigidity from the network.

The layered structure of SDN architecture, as shown in Fig. 1 has three major planes such as the data plane, the control plane, and the management plane. The data plane contains physical network elements, which form the path for data transmission. The control plane has a Network Operating System (NOS), also referred to as a controller, which generates the flow rule table for devices in data plane. These rules and policies are designed in the management plane of SDN architecture. The communication between these planes is established by using well-defined Application Programmable Interfaces (APIs). These interfaces are divided into southbound, northbound, eastbound, and westbound APIs. The communications between the control plane and the data plane is implemented through the southbound API, which enables flow installation and configuration of devices. The control plane and the management plane use northbound API to provide programmability in SDN. Inter-controller communication of SDN domains is established using eastbound API, whereas westbound API is responsible for the legacy domain to SDN domain communication. The detail of these interfaces can be found in some literatures.

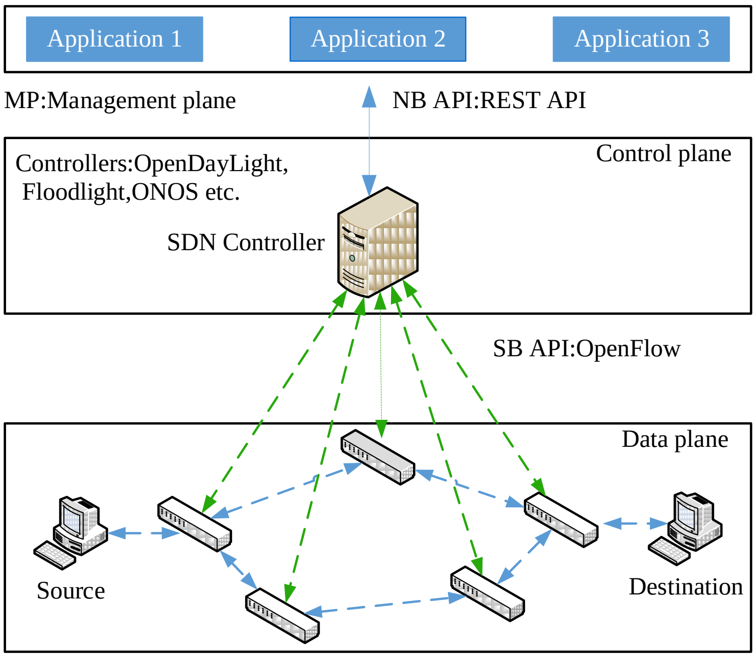


Fig. 1. The SDN Architecture [3]

3.2. The Link Layer Discovery Protocol

The decoupling between the control plane and the data plane introduced by SDN allows operators to employ remarkably cheap but very fast hardware to forward packets, moving the control logic to the much smarter controller. The controller plays the role of an operating system of the network. One of fundamental functions that a controller must offer is an accurate, nearly real time view of the network topology. This function is known as the topology discovery. The Link Layer Discovery Protocol (LLDP) [30] is a standard method of the network topology discovery in SDN. Fig. 2 shows the principle that how LLDP works. To discover the unidirectional link $s_1 \rightarrow s_2$, the controller encapsulates a LLDP packet in a Packet-out message and sends it to s_1. The Packet-out contains instruction for s_1 to send the LLDP packet to s_2 via port p_1. When s_2 receives the LLDP packet via port p_2, s_2 encapsulates it as a destination switch in a Packet-in message and sends it back to the controller. The controller receives the LLDP packet and concludes that there is a unidirectional link from s_1 to s_2. The same process is performed to discover the opposite direction $s_2 \rightarrow s_1$ as well as all other links in the network. After all switches perform such operations, the controller will obtain the network topology information of the entire network. However, the network topology will dynamically change incurred by switches leave and join the network. So the controller needs to periodically repeat the process described in Fig. 2.

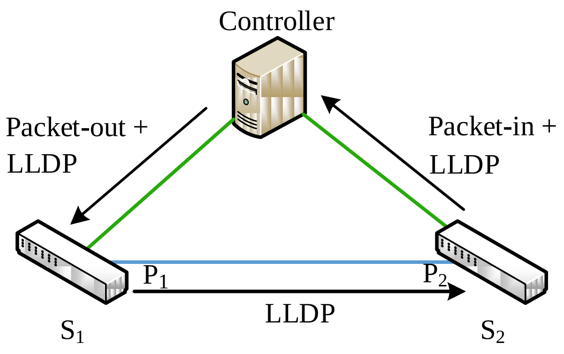


Fig. 2. The link discovery process [30]

3.3. Blockchain and PoW

Blockchain technology has been applied in many areas [18-19]. Blockchain is a system that is composed of nodes, communicating with each other through a protocol. A node can be a physical machine or a virtual machine. The IP address is used to identify the node in the Blockchain network. The public key is used as an user identification in the network. The private key is generally used for signing on message transmitted on the network. As a result, each user can log in from any node in the system. The consistency of data stored on Blockchain must be guaranteed on the entire network and can be achieved by some consensus algorithm such as PoW, PoS and so on [36]. And data on the Blockchain network is digitally signed to guarantee authenticity and accuracy properties. Blockchain technology can ensure an immutable storage and a fraud protection property. The work mechanism of the Blockchain network is shown in Fig. 3. The transaction data is stored in a specific data structure called “block” in Blockchain. The blocks generated during the transaction process are linked together via the cryptographic hash function to form a chain of blocks. That is to say, each block inside the Blockchain stores a hash value of the previous block. Thus, the chain of blocks is grouped or linked in a chronological order. As a result, the data that stored on the

Blockchain won’t be modified without cooperation of all nodes inside the system. So, the mechanism provides a proof-tamper feature.

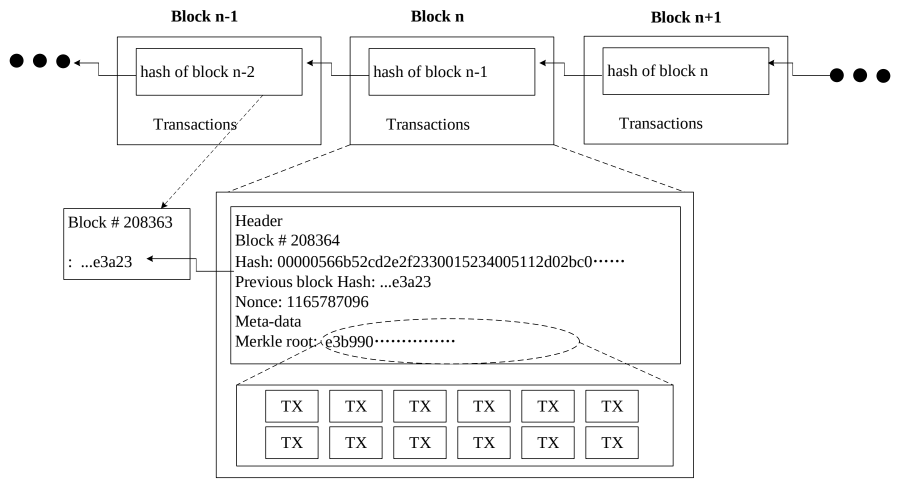


Fig. 3. The Blockchain and block structure [18]

Blockchain technology has been applied in many areas [18-19]. Blockchain is a system that is composed of nodes, communicating with each other through a protocol. A node can be a physical machine or a virtual machine. The IP address is used to identify the node in the Blockchain network. The public key is used as an user identification in the network. The private key is generally used for signing on message transmitted on the network. As a result, each user can log in from any node in the system. The consistency of data stored on Blockchain must be guaranteed on the entire network and can be achieved by some consensus algorithm such as PoW, PoS and so on [36]. And data on the Blockchain network is digitally signed to guarantee authenticity and accuracy properties. Blockchain technology can ensure an immutable storage and a fraud protection property. The work mechanism of the Blockchain network is shown in Fig. 3. The transaction data is stored in a specific data structure called “block” in Blockchain. The blocks generated during the transaction process are linked together via the cryptographic hash function to form a chain of blocks. That is to say, each block inside the Blockchain stores a hash value of the previous block. Thus, the chain of blocks is grouped or linked in a chronological order. As a result, the data that stored on the Blockchain won’t be modified without cooperation of all nodes inside the system. So, the mechanism provides a proof-tamper feature.

3.4. Proof of Work

Blockchain is a key technology to build a distributed trust in the environment that users don’t trust each other and there doesn’t exist a Trust Third Party (TTP). In Blockchain network, the consensus scheme ensures the consistency of data stored on the Blockchain. More recently, some consensus schemes for Blockchain have been proposed and most of them are based on three basic algorithms that often used in a distributed network, such as Proof of Work (PoW) [35], Proof of Stake (PoS) and Direct Acyclic Graph (DAG). A comprehensive performance comparison is done among them in [36].

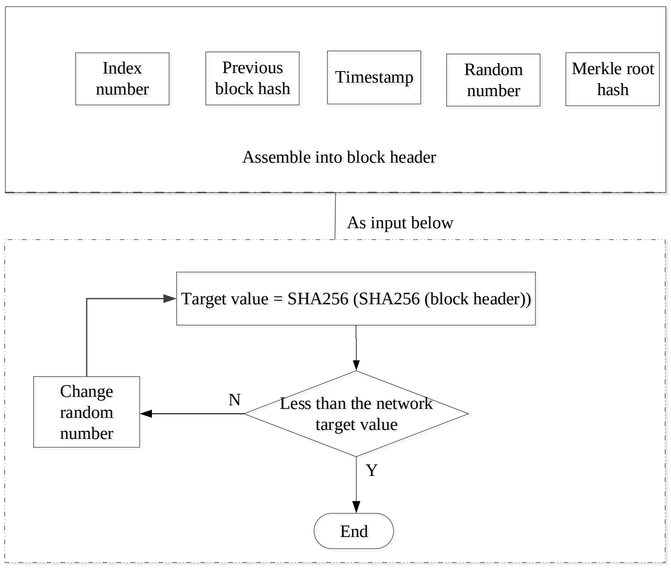


Fig. 4. Proof of Work

PoW [35] used in Bitcoin is the most classical consensus algorithm in the Blockchain. The PoW involves a scanning of a hash value that computed by using a hash algorithm such as SHA-256. The hash value begins with a string of 0 bits. The average workload is exponent in the number of 0 bits required and can be verified by executing a single hash. The PoW is implemented by incrementing a nonce in the block until a hash value that contains the required number of 0 bits in the block's hash. Fig. 4 shows how the PoW works. Once the computed value satisfies the requirement of PoW, the block cannot be changed without re-executing the work. As subsequent blocks are chained to the new generated block, modifying a block means regenerating all the blocks after the modified block. So the core idea of PoW used in Blockchain is that miners use their computing power to compete the hashing operation. The winner who first finds the hash value lower than the announced target has the right to insert a new block into the blockchain and get a certain amount of reward.

3.5. Keyless Signature Instructure

The Keyless Signature Instructure (KSI) [34] is a globally distributed system for providing digital signature services. KSI is an alternative solution to traditional PKI signature. It has some benefits and has been payed widely attention. It can detect the change status of digital assets and submit this information for further audit and investigation. A mechanism with multiple signatures can be obtained in KSI. That is to say, multiple documents can be signed together at once. The signing process includes the following three steps. *Hash*: a hash value of the data or file generated by the client will be calculated; *Aggregation*: The gateway layer collects and processes the hash values that comes from the clients, aggregates them into a Merkle tree, and sends the generated root hash value to the aggregation layer. The aggregation layer server processes the root hash value generated and sent by the gateway layer, and adds it to the Merkle tree. Finally, the generated root hash value will be transferred to the core layer; *Release*: a permanent hash tree will be created according to the first three hash values of the aggregation tree collected each time and it is released as a trust anchor.

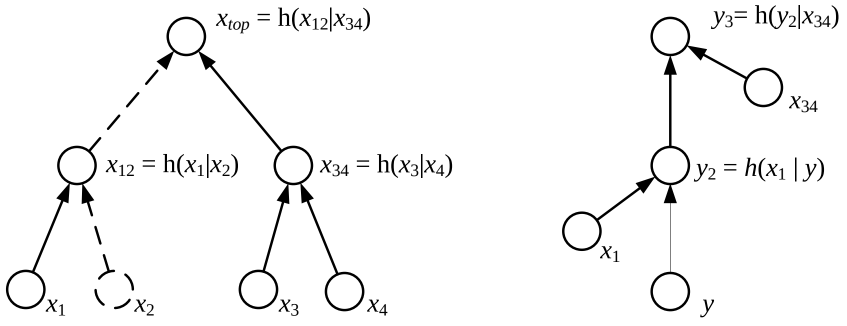

x_{top} = h(x_{12}|x_{34})
x_{12} = h(x_1|x_2)
x_{34} = h(x_3|x_4)
x_1
x_2
x_3
x_4
y_3= h(y_2|x_{34})
x_{34}
$y_2 = h(x_1 \mid y)$
x_1
y

Fig. 5. Hash Tree and Hash Calendar. [34]

Hash Trees: Hash-tree aggregation process described before was first introduced in [38]. In hash-tree time-stamping scheme, a one-way hash function is used to convert a list of data or files into a fixed length hash value that is generally associated with time. A signature token generated by the service according to a hash of a document from client is considered as a proof that the data sent by the client existed at the given time and that the request was received through a specific access point. All received requests are aggregated together into a large hash tree; and the top of the tree is fixed and retained for each second as shown in Fig.5. The signature token contains data for reconstructing a path through the hash tree—starting from a signed hash value (a leaf) to the top hash value. For example, to verify a token y in the place of x_2 (Fig. 5), a concatenation operation is firstly done between y and x_1 (retained as a part of the signature token) and then a hash value $y_2 = h(x_1 \mid y)$ is calculated and is used as the input of the next hash step, the process will be end when it reach the top hash value, i.e. $y_3 = h(y_2|x_{34})$ in the example case. If $y_3 = x_{top}$ then it is safe to prove that y was in the original hash tree.

Hash Calendar: These top hash values obtained in each round are linked together to generate a globally unique hash tree (The hash tree is called a hash calendar in [34])—so that new leaves are added only to one side of the tree. Time value is encoded as the shape of the calendar—the modification of which would be evident to other users. However, the top hash of the calendar is required to periodically publish in widely witnessed media. There is a deterministic algorithm to compute the top hash of the linking hash tree, giving a distinct top level hash value at each second. Also there is an algorithm to extract time value from the shape of the linking hash tree for each second, giving a hard-to-modify time value for each issued token.

4. BCSDN Framework

The BCSDN architecture proposed in this paper is a distributed multi-controller architecture. In the control plane of BCSDN, the controllers collect the link status information from each switch that joined to the network, by using the link discovery protocol LLDP during the link discovery phase. A switch will package the link information in a Packet_in packet during the link discovery, and then it will submit the Packet_in packet to a controller. In BCSDN, the submission is considered as a transaction process in the Blockchain network. The consensus algorithm such as PoW is used to elect a main controller from these controllers. The selected main controller that plays a role of a miner verifies the transaction and aggregates all of hash values to generate a Merkle tree according to KSI algorithm. Finally, it will generate a block of the root hash value and storage on the Blockchian network, and then it will issue signature to each switch according to KSI signature rule. Each block is related to a hash calendar. Thus, the chain of blocks records and represents the dynamical change process of the network topology. The main controller will issue the latest network topology information collected from the network to other controllers so that the scheme ensures the consistency of the network view among controllers. When a switch in the network needs to forward a data, interaction will be performed between the switch and the controller that directly connected with the switch to request the flow rule table. The controller sends the latest flow rule table after verifying the signature owned by the switch. So, our BCSDN framework is shown in Fig. 6 and it is consisted of the following 5 components: The network topology generation, Blockchain establishment, the selection of the main controller, the signature generation and signature verification.

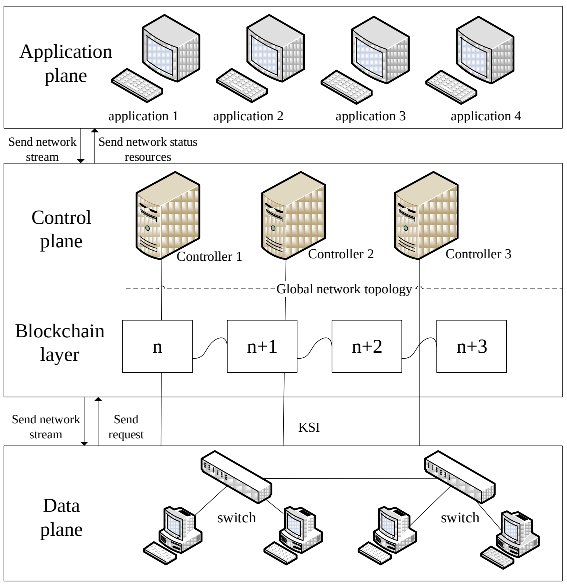


Fig. 6. The Entire Framework of BCSDN

4.1. The Network Topology Generation

In BCSDN, the standard link layer discovery protocol LLDP introduced in section 3.2 is used to collect the network link state information. The SDN controller initiates the link discovery process. The process consists of the following 4 steps. (1) The SDN controller periodically sends a LLDP packet Packet_out packet to all switches that connected with it. (2) Once a switch receives the Packet_Out packet from the SDN controller, it will broadcast the Packet_out packet to all of devices that connected with the switch via all of its ports. （3) In our BCSDN, we assume that the neighboring switches are an OpenFlow switch. That is to say, the switch have no a special flow rule entry for processing LLDP messages, so they will send a LLDP packet Packet_in packet to the controller connected with them. (4) After the controller receives a Packet-In packet, it will analyze the data packet and save the link information between the two switches in its link discovery table and calculate a hash value of message in the Packet-In packet by using SHA256. The algorithm is described in Algorithm 1.

Algorithm1: Hash the Link Information

```
Input: Packet_out{}
Output: temptxList{}
1：while a switch receives packet_out{} message do
2：    forward the message to neighboring switches
3：    neighboring switches  sends the packet_in to the controller
4：        temptxList{} ← sha256(packet_in )
5：return temptxList{}
```

4.2. Blockchain Establishment

Algorithm2: Merkle Tree Construction

```
Input: temptxList{}
Output: root
1：while newTxList.size() != 1 do
2：    index = 0
3：    while index < tempTxList.size() do
4：        left ← tempTxList.(index)
5：        index++
6：        right ← " "
7：        if  index != tempTxList.size() then
8：            right = tempTxList(index)
9：        newTxList{} ← SHA256(left , right)
10：       index++
11：root ← newTxList(0)
12：return root
```

In BCSDN, the main controller will generate a Merkle tree according to the KSI scheme during the link discovery process. The main controller will save the hash value of the root node as a block on the Blockchain maintained by these controllers according to the principle of Blockchain. A new block on Blockchain is set up as shown in Fig. 6. That is to say, an update (a hash calendar) of the network topology will generate a new block on the Blockchain. That is to say, the Merkle tree locally represents the current network topology information. The construction algorithm of the Merkle tree is shown in Algorithm 2.

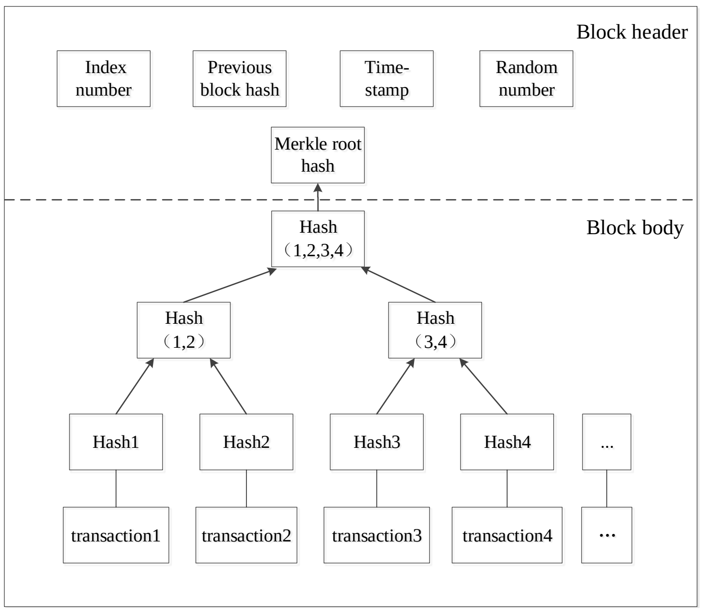


Fig. 6. A block generation on Blockchain

4.3. Selection of a Main Controller

In order to solve the single-point failure problem incurred by a single controller in SDN, a multi-controller architecture is adopted in our BCSDN framework. These controllers are deployed in physically distributed and logically centralized manner. The PoW introduced in section 3.4 is used for selecting a main controller from these controllers. The algorithm is described in Algorithm 3.

The main controller just selected will play a role of a miner in the Blockchain network and issue the topology information to all of controllers. So, our novel BCSDN can ensure the consistency of the network topology information on these different controllers.

Algorithm3: Proof of Work

```
Input: index, rootHex, time, previousBlockHash, random, TargetValue
Output: Block
1: Block ← index + rootHex + time + previousBlockHash + random
2: while true do
3:     if SHA256(SHA256(Block)) >= TargetValue then
4:         random ++
5:         Block ← index + rootHex + time + previousBlockHash + random
6:     else
7:         break
8:     end if
9: end while
10: return Block
```

4.4. The Signature Generation

The main controller selected in PoW process will manage the entire network. The main controller will use the KSI scheme to generate and issue a signature to each switch that submitted the correct link information to it. The signature information generated by the main controller is a concatenation of these hash values on nodes which is located on the Merkle tree. These nodes together form a path from a leaf node that represents information of a link to the root of the Merkle tree. So, this signature means that the switch sent a Packet_In packet containing the link information to construct the Merkle tree that recorded the current network topology information. The signature process is shown in Algorithm 4.

Algorithm4: Signature Generation

```
Input: CalculationPath, node
Output: HashSignature{}
Initialization: HashSignature{}= ∅
1: if the node is in CalculationPath and is a leaf
2:     HashSignature{} ← node
3: if the node is in CalculationPath
4:     traversing left and then right
5: otherwise HashSignature{} ← node
6: return HashSignature{}
```

4.5. Signature Verification

When a switch need to forward data for an end user, it firstly checks if there is a matching entry in a flow rule table. If the check fails, a forwarding request event is generated and then the switch sends the request with signature the main controller issued to it to the controller connected with it. When the controller receives the request, it verifies the signature of the switch according to KSI algorithm. If the verification is successful, the controller transfers the latest flow rule table to the switch. Otherwise, it drops the request and don’t response to the switch. The algorithm of signature verification is described in Algorithm 5. The signature scheme used in BCSDN can efficiently authenticate the switch and prevent some attacks such as spoofing incurred by malicious switches controlled by adversary.

Algorithm5: Signature Verification

```
Input: HashSignature{node 0,node 1…node i}, root
Output: result
Initialization: HashNode = ∅
1:  HashNode ← node 0
2:  for (n=1, n<=i, n++)
3:    HashNode= sha256(HashNode, node n)
4:  if  HashNode == root  then
5:      result ← verify successfully
6:      break
7:  else
8:     result ← verify failed
9:  end if
10: return  result
```

5. Security Analysis

In this section, we informally discuss the security issues solved in our BCSDN. BCSDN can efficiently solve the single point of failure, the view consistency of multi-controller SDN network and the authentication of the interaction between a controller and a switch.

***Proposition* 1**: BCSDN can solve the single point of failure.

Proof: In BCSDN, the multi-controller architecture is used to solve the single point of failure. That is to say, a logically centralized and physically distributed multi-controller framework is adopted in the control plane of SDN. The selected main controller manages the entire network. When the main controller shut down because of some reason, the re-elected main controller will take over the network management task. Therefore, the multi-controller architecture can conquer the single point of failure, improve the processing capacity of the control plane, and also ensure the reliability of the network management.

***Proposition* 2**: BCSDN can ensure the consistency of the network topology on the different controllers.

Proof: In BCSDN, the mechanism based on Blockchain is implemented in the control plane. The dynamic change of the network topology will be recorded in Blockchain. These features of Blockchain such as proof-tamper, auditable, distributed storage and so on will ensure that the network topology information stored in the Blockchain is correct and won't be modified and also guarantee that the network topology information stored on the different controllers is consistent.

***Proposition* 3**: The communication between a controller and a switch can be authenticated.

Proof: In BCSDN, KSI is used when a controller collect the topology information of the network. That is to say, the controller will generate a signature for each switch according to the KSI mechanism in link discovery process. The signature is a proof that proves the switch ever took part in the link discovery process in the corresponding network topology. That is to say, each link (that submitted by the switch) in the network topology is a legal link. The KSI scheme ensures that the signature that the controller sent to each switch can't be forged and these signatures information are saved on Blockchain. When a switch that needs to forward data requests the flow rule table from a controller, the controller will verify the signature generated in link discovery process and owned by the switch. So, using of KSI can ensure the authenticated communication between a controller and a switch.

6. BCSDN Implementation

6.1. BCSDM simulation Implementation

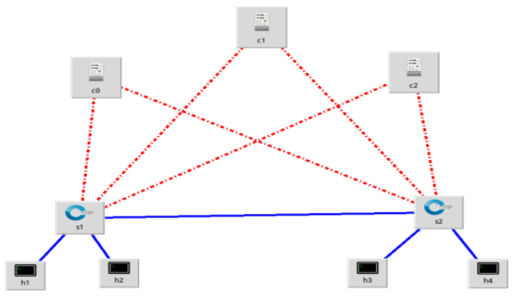
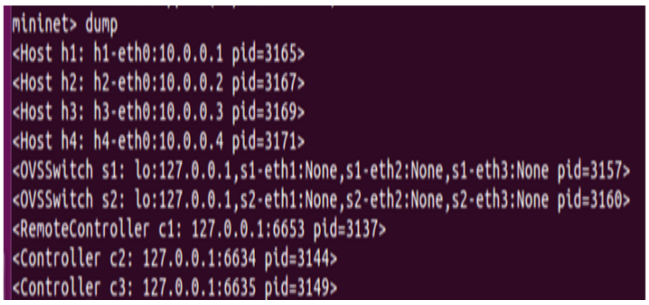


Fig. 7. The Simulated Network Topology.

In this section, we implement our BCSDN architecture by using simulation method. The network emulator Mininet on Ubuntu 16.04 system is used to simulate a custom topology of a SDN network. The Floodlight controller is used to establish a multi-controller architecture for SDN to manage the entire network. We illustrate the operation of the BCSDN network by a simply network topology as shown in Figure 7. Three Floodlight controllers are used to establish a multi-controller architecture. These controllers together form a simply p2p network and are used to manage and maintain a Blockchain network. The main controller is selected by using the PoW alogrithm. We implement the network architecture instance of BCSDN in the network simulation platform Mininet. These controllers in BCSDN periodically collect the network link information by the link discovery protocol LLDP introduced in 3.2 section. The main controller generates the signature corresponding to each link according the Merkle tree that presents the current network topology. The network state information is recorded on the Blockchain as a distributed ledger. The simulation topology includes 2 OpenFlow switches and 4 hosts connected to these two switches respectively. Built-in Mininet network commands such as "ping" and so on can be used to test the correctness, feasibility and overhead of BCSDN. The details of each node in the simulation topology are described as shown in Figure 8. In Fig. 9a, the information from the log file of the node displays the block creation and the collection of the network topology. In Fig. 9b, the signature verification result is shown when the switch applies for the flow rule table from the controller.



```
mininet> dump
<Host h1: h1-eth0:10.0.0.1 pid=3165>
<Host h2: h2-eth0:10.0.0.2 pid=3167>
<Host h3: h3-eth0:10.0.0.3 pid=3169>
<Host h4: h4-eth0:10.0.0.4 pid=3171>
<OVSSwitch s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None,s1-eth3:None pid=3157>
<OVSSwitch s2: lo:127.0.0.1,s2-eth1:None,s2-eth2:None,s2-eth3:None pid=3160>
<RemoteController c1: 127.0.0.1:6653 pid=3137>
<Controller c2: 127.0.0.1:6634 pid=3144>
<Controller c3: 127.0.0.1:6635 pid=3149>
```

Fig. 8. Information of the Network Nodes.

```
2020-08-11 09:04:21.988 INFO  [n.f.l.i.LinkDiscoveryManager]
Received LLDP packet on sw 00:00:00:00:00:00:00:01, port 2
The first block：     Block{
                        index=0,
                        rootHex='IFmBkeOfebmNHm0qTf508QsMrfPTflqMt5yKph8LsYU=',
                        time=1597161862025,
                        previousBlockHash=0,
                        random=945}
```

```
The second block：  Block{
                       index=1,
                       rootHex='65Bm8Zbb69XU3pWHDusPh8mc7sicAm9TmRqJoFlwcTs=',
                       time=1597161862064,
                       previousBlockHash=NebGApj7T6pIv7Ho87IbMNDdfPLOrE3ix1wU7D0mjm4=,
                       random=388}
```

a. Block information

```
2020-08-11 09:05:46.865 INFO  [n.f.f.Forwarding] sign pass
```

b. The Signature Verification

Fig. 9. The results of the Network Simulation.

6.2. Performance Analysis

The performance of BCSDN is analyzed under different network scale in this section. We compare performance metrics such as the network convergence time, network throughput and the response time between the singe controller solution and our BCSDN. The Floodlight controller is used in the experimental analysis.

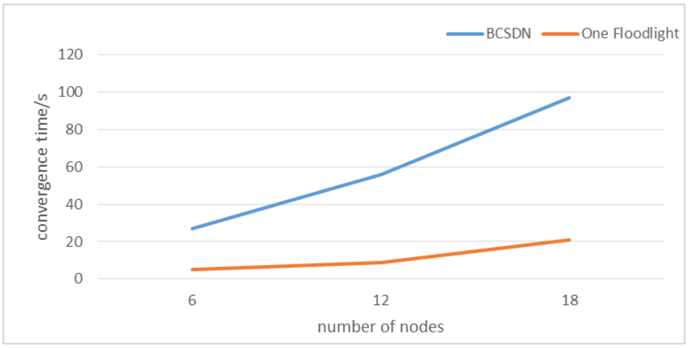


Fig. 10. The Network Convergence time

Fig.10 shows the convergence time of these two simulated networks that one only uses a single controller and another uses our BCSDN solution. It can be clearly seen that these different networks can converge successfully in three different network sizes and obtain the entire network topology. BCSDN needs to set up Blockchain network according to the change of the network topology. So, the convergence time is higher than the solution used the single controller. But, we can see that the curve of the convergence time in BCSDN will become smooth as the network size increases. That is to say, BCSDN is better than the network that there exists only a single controller in large scale network.

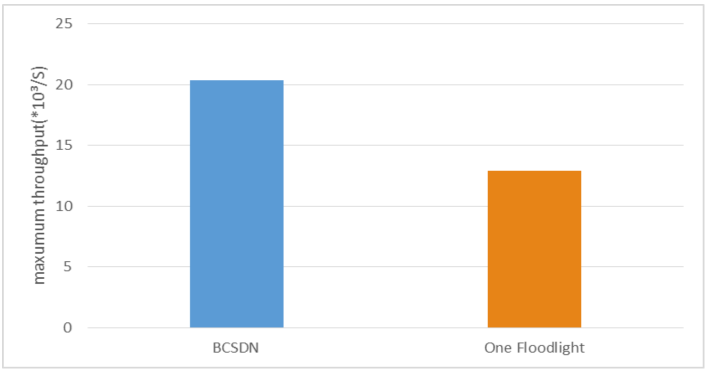
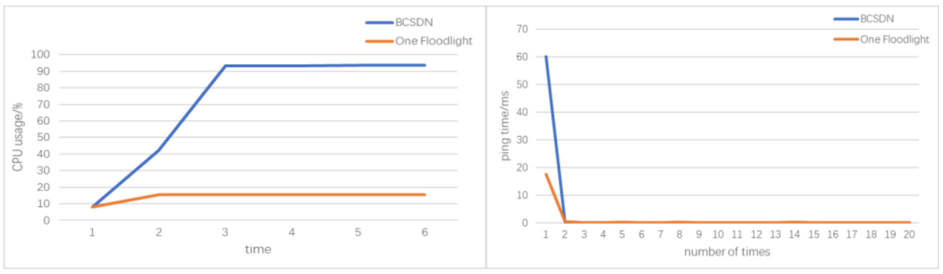


Fig. 11. Throughput Comparison Between the Sing Controller network and BCSDN

The network throughput is compared between BCSDN and the network with the single controller as shown in Figure 11. In BCSDN, three controllers are used. the processing ability is indeed stronger than the network with a single controller. This shows that our multi-controller network architecture is more excellent than the single controller network architecture.



a. CPU usage b. The Response Time

Fig. 12. Comparison of CPU usage and the Response Time

In addition, the CPU usage rate and the response time are analyzed in these two solutions as shown in Fig.12. Clearly, because using of Blockchian, the CPU usage rate in BCSDN is higher than the solution that used a single controller as Fig. 12a. However, the response time is tested by using performing 20 ping operations. As shown in Figure 12b, it can be seen that the ping response time of BCSDN and a single controller network is almost the same except that the time of the first ping. That is to say, BCSDN can meet the requirements of network rapid processing.

7. Summary

In this paper, a security framework for SDN BCSDN is proposed by integrating Blockchain and KSI in this paper. The BCSDN adopts a physically distributed and logically centralized multi-controller architecture. In BCSDN, LLDP protocol is used to obtain the topology information of the network, and the dynamic change of the network topology is recorded on Blockchain. The main controller selected according to PoW play a role that manage the entire network. So, BCSDN can efficiently solve the single point of failure in single-controller architecture. In addition, using of Blockchain scheme ensures the consistency of the topology information on different controllers. Using of KSI is used to authenticate the communication between a controller and a switch. The correctness, reliability and feasibility are verified by an emulation method in mininet emulation platform in this paper. We also simply analyze security attributes and performance of the BCSDN framework. We will further improve our solution in the future work.

Acknowledgments. This work is supported by NSFC No. 61461027; Gansu province science and technology plan project under grant No. 20JR5RA467; Innovation Promotion Education Fund of Ministry of Education No. 2018A05003.

References

1. Rishikesh Sahay, Weizhi Meng, Christian D. Jensen.: The application of Software Defined Networking on securing computer networks: A survey [J]. Journal of Network and Computer Applications, Vol. 131, 89-108. (2019)
2. Y. Zhao, Y. Li, X. Zhang, G. Geng, W. Zhang and Y. Sun.: A Survey of Networking Applications Applying the Software Defined Networking Concept Based on Machine Learning [J], IEEE Access, Vol. 7, 95397-95417. (2019)
3. Ali, J.; Lee, G.-M.; Roh, B.-H.; Ryu, D.K.; Park, G.: Software-Defined Networking Approaches for Link Failure Recovery: A Survey [J], Sustainability, Vol. 12, No. 10, 4255. (2020)
4. Sanjeev Singh, Rakesh Kumar Jha.: A Survey on Software Defined Networking: Architecture for Next Generation Network [J]. Journal of Network and Systems Management, Vol. 25, 321-374. (2016)
5. Hu T, Guo Z, Yi P, et al. Multi-controller based software-defined networking: A survey [J]. IEEE Access, Vol. 6, 15980-15996. (2018)
6. Heng Zhang, Zhiping Cai, Qiang Liu, Qingjun Xiao, Yangyang Li, Chak Fone Cheang.: A Survey on Security-Aware Measurement in SDN [J], Security and Communication Networks, vol. 2018, 14 pages. (2018)
7. Y. Liu, B. Zhao, P. Zhao, P. Fan and H. Liu.: A survey: Typical Security Issues of Software-Defined Networking [J], China Communications, vol. 16, no. 7, 13-31. (2019)
8. Juan Camilo Correa Chica, Jenny Cuatindioy Imbachi, Juan Felipe Botero Vega.: Security in SDN: A Comprehensive Survey [J], Journal of Network and Computer Applications, Vol. 159, 102595. (2020)
9. Tao Han, Syed Rooh Ullah Jan, Zhiyuan Tan, et. al.: A Comprehensive Survey of Security Threats and Their Mitigation Techniques for Next-generation SDN Controllers [J], Concurrency and Computation：Practice and Experience, Vol. 32. (2020)
10. Tao Hu, Peng Yi, Zehua Guo, Julong Lan, Yuxiang Hu.: Dynamic slave controller assignment for enhancing control plane robustness in software-defined networks [J], Future Generation Computer Systems, Vol. 95, 681-693. (2019)
11. Madhukrishna Priyadarsini, Joy Chandra Mukherjee, Padmalochan Bera, Shailesh Kumar, A. H. M. Jakaria, M. Ashiqur Rahman. An adaptive load balancing scheme for software-defined network controllers, Computer Networks, Vol. 164 (2019)
12. A. J. Gonzalez, G. Nencioni, B. E. Helvik and A. Kamisinski.: A Fault-Tolerant and Consistent SDN Controller [C], In Proceeding of 2016 IEEE Global Communications Conference (GLOBECOM), Washington, DC, 1-6 (2016)
13. Das, R.K., Pohrmen, F.H., Maji, A.K. et al.: FT-SDN: A Fault-Tolerant Distributed Architecture for Software Defined Network [J]. Wireless Personal Communication, Vol. 114, 1045–1066. (2020).
14. E. Sakic, N. Đerić and W. Kellerer. : MORPH: An Adaptive Framework for Efficient and Byzantine Fault-Tolerant SDN Control Plane [J], IEEE Journal on Selected Areas in Communications, Vol. 36, No. 10, 2158-2174. (2018)
15. K. ElDefrawy and T. Kaczmarek. : Byzantine Fault Tolerant Software-Defined Networking (SDN) Controllers [C], In Proceeding of 2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC), Atlanta, GA, 208-213. (2016)
16. Gao J., Wu J. X.,Hu Y. X,et al. Research on Anti-attack of Software-Defined Network Control Surface Based on Byzantine Fault Tolerance [J]. journal of Computer Applications, Vol. 37, No. 8, 2281-2286. (2017) (in Chinese).
17. Luis Guillen, Hiroyuki Takahira, Satoru Izumi, Toru Abe, Takuo Suganuma.: On Designing a Resilient SDN C/M-Plane for Multi-Controller Failure in Disaster Situations [J], IEEE Access, Vol. 8, 141719-141732. (2020)

18. Dharmin Dave, Shalin Parikh, Reema Patel, Nishant Doshi.:A Survey on Blockchain Technology and its Proposed Solutions [J], Procedia Computer Science, Vol. 160, 740-745. (2019)
19. Gamage, H.T.M., Weerasinghe, H.D. & Dias, N.G.J.: A Survey on Blockchain Technology Concepts, Applications, and Issues [J]. SN Computer Science, Vol. 1, 114. (2020).
20. TALAL ALHARBI.: Deployment of Blockchain Technology in Software Defined Networks: A Survey [J], IEEE Access, Vol. 8, 9146-9156. (2020)
21. S. R. Basnet and S. Shakya.: BSS: Blockchain security over software defined network [C], In Proceeding of the 2017 International Conference on Computing, Communication and Automation (ICCCA), Greater Noida, 720-725. (2017)
22. Huo, L., Jiang, D., Qi, S. et al.: A Blockchain-Based Security Traffic Measurement Approach to Software Defined Networking [J]. Mobile Networks and Applications, (2020)
23. Bo Zhao, Yifan Liu, Xiang Li, Jiayue Li, Jianwen Zou.: TrustBlock: An adaptive trust evaluation of SDN network nodes based on double-layer blockchain [J], PLoS One, Vol. 15, No. 3, e0228844. (2019).
24. Hien Do Hoang, Phan The Duy, Van Hau Pham.: A Security-Enhanced Monitoring System for Northbound Interface in SDN using Blockchain [C], In Proceedings of the Tenth International Symposium on Information and Communication TechnologyDecember, NY, USA, 197–204. (2019)
25. C. Xue, N. Xu and Y. Bo. : Research on Key Technologies of Software-Defined Network Based on Blockchain [C], In Proceeding of 2019 IEEE International Conference on Service-Oriented System Engineering (SOSE), San Francisco East Bay, CA, USA, 239-2394. (2019)
26. A. Derhab, M. Guerroumi, M. Belaoued, O. Cheikhrouhou. : BMC-SDN: Blockchain-Based Multicontroller Architecture for Secure Software-Defined Networks, Wireless Communications and Mobile Computing, vol. 2021, Article ID 9984666, 12 pages. (2021)
27. A. Rahman, M. K. Nasir, Z. Rahman, A. Mosavi, S. S. and B. Minaei-Bidgoli. DistBlockBuilding: A Distributed Blockchain-Based SDN-IoT Network for Smart Building Management [J], IEEE Access, vol. 8, 140008-140018. (2020)
28. D. Zlate, F. Sonja, M. Anastas, T. Vladimir. : Real time availability and consistency of health-related information across multiple stakeholders: A blockchain based approach [J], Computer Science and Information Systems, Vol. 18, No. 3, 927-955. (2021)
29. Abbas Yazdinejad, Reza M. Parizi, Ali Dehghantanha, Kim-Kwang Raymond Choo.: P4-to-blockchain: A secure blockchain-enabled packet parser for software defined networking [J], Computers & Security, Vol. 88. (2020)
30. W.-Y. Huang, T.-Y. Chou, J.-W. Hu, and T.-L. Liu. : Automatical end to end topology discovery and flow viewer on SDN [C], In Proceeding 2014 28th International Conference on Advanced Information Networking and Applications Workshops, Victoria, BC, 910-915. (2014)
31. Mowla Nishat I, Doh Inshil, Chae Kijoon,: CSDSM: Cognitive switch-based DDoS sensing and mitigation in SDN-driven CDNi word [J], Computer Science and Information Systems, Vol. 15, No. 1, 163-185. (2018)
32. Dierks T, Rescorla E. The transport layer security (TLS) protocol version 1.2. RFC 5246, 1-104. (2008).
33. A. Azzouni, N. T. Mai Trang, R. Boutaba and G. Pujolle. : Limitations of openflow topology discovery protocol [C], In Proceeding of 2017 16th Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net), Budva, 1-3. (2017)
34. Buldas A., Kroonmaa A., Laanoja R.: Keyless Signatures' Infrastructure: How to Build Global Distributed Hash-Trees [C], Nordic Conference on Secure IT Systems, Lecture Notes in Computer Science, Vol. 8208, 313-320.(2013)
35. Bin Cao, Zhenghui Zhang, Daquan Feng, et. al. : Performance analysis and comparison of PoW, PoS and DAG based blockchains [J], Digital Communications and Networks, Vol. 6, 480-485. (2020)

36. Satoshi Nakamoto.: Bitcoin: A Peer-to-Peer Electronic Cash System, https://bitcoin.org/en/bitcoin-paper.
37. R. L. S. de Oliveira, C. M. Schweitzer, A. A. Shinoda and Ligia Rodrigues Prete. : Using Mininet for emulation and prototyping Software-Defined Networks [C], In Proceeding of the 2014 IEEE Colombian Conference on Communications and Computing (COLCOM), Bogota, 1-6. (2014)
38. Merkle, R.C.: Protocols for public-key cryptosystems [C]. In Proceedings of the 1980 IEEE Symposium on Security and Privacy, Oakland, CA, USA, 122–134 (1980)

Xian Guo, is an associate professor of School of Computer and Communication, Lanzhou University of Technology. He is a visiting scholar at University of Memphis. He received MS and PhD in Lanzhou University of Technology, China, in 2008 and 2011, respectively, and BS in Nothwest Normal University. His current research interests include network and information security, cryptographic, and blockchain. E-mail: iamxg@ 163.com.

Chen Wang, is currently a master student at Computer and Communication School of Lanzhou University of Technology. He received his Bachelor degree from Lanzhou University of Technology in 2017, and started his master studying in 2018. His research interests are Software Defined Networking, security of wireless network, blockchain.

Laicheng Cao, is a professor of School of Computer and Communication, Lanzhou University of Technology. He received MS in Lanzhou University, China, in 2004. His current research interests include network and information security, cryptography etc.

Yongbo Jiang, is a lecturer of School of Computer and Communication, Lanzhou University of Technology. He received MS and PhD in Xidian University, China, in 2008 and 2013, respectively. His current research interests include network and information security, information-centric networking etc.

Yan Yan, is an associate professor of School of Computer and Communication, Lanzhou University of Technology. She received MS and PhD in Lanzhou University of Technology, China, in 2005 and 2018, respectively. Her current research interests include machine learning, privacy computing, and blockchain.

Received: February 02, 2021; Accepted: December 27, 2021.