Hindawi
Security and Communication Networks
Volume 2022, Article ID 7294638, 16 pages
https://doi.org/10.1155/2022/7294638

Research Article

BlockREV: Blockchain-Enabled Multi-Controller Rule Enforcement Verification in SDN

Ping Li,[1,2] Songtao Guo,[3] Jiahui Wu,[1] and Quanjun Zhao[1]

[1]*College of Electronic and Information Engineering, Southwest University, Chongqing 400715, China*
[2]*College of Mathematics and Statistics, North China University of Water Resources and Electric Power, Zhengzhou 450046, China*
[3]*College of Computer Science, Chongqing University, Chongqing 400044, China*

Correspondence should be addressed to Ping Li; liping_sx@ncwu.edu.cn

Received 29 September 2021; Accepted 17 December 2021; Published 11 January 2022

Academic Editor: Fulvio Valenza

Compared with the classical structure with only one controller in software-defined networking (SDN), multi-controller topology structure in SDN provides a new type of cross-domain forwarding network architecture with multiple centralized controllers and distributed forwarding devices. However, when the network includes multiple domains, lack of trust among the controllers remains a challenge how to verify the correctness of cross-domain forwarding behaviors in different domains. In this paper, we propose a novel secure multi-controller rule enforcement verification (BlockREV) mechanism in SDN to guarantee the correctness of cross-domain forwarding. We first adopt blockchain technology to provide the immutability and privacy protection for forwarding behaviors. Furthermore, we present an address-based aggregate signature scheme with appropriate cryptographic primitives, which is provably secure in the random oracle model. Moreover, we design a verification algorithm based on hash values of forwarding paths to check the consistency of forwarding order. Finally, experimental results demonstrate that the proposed BlockREV mechanism is effective and suitable for multi-controller scenarios in SDN.

1. Introduction

Software-defined networking (SDN) is more agile by means of network programming [1]. With the development of edge computing and artificial intelligence (AI) technology, AI-enabled SDN provides users with a variety of applications [2]. Compared with the classical SDN network with only one controller, multi-controller framework can provide a new type of cross-domain forwarding network architecture with multiple centralized controllers and distributed forwarding devices and has more benefits of flexibility and scalability. It can overcome some drawbacks of classical SDN, such as weak computing power, limited scalability, and high load of the single controller. Clearly, it is very important to verify the correctness of cross-domain forwarding rule execution by using cryptology primitives or statistical knowledge, which is referred to as rule enforcement verification. It ensures the validity of cross-domain forwarding behaviors and maintains perfect network status, so as to offer better service-level agreements (SLAs) for clients and meet the needs of customized services.

However, multi-controller rule enforcement verification technology still faces some security challenges. First, it lacks the trust among controllers. Forwarding verification in each domain is managed by its own controller. If the controller is compromised, the entire network will be subject to single-point failure attack. Adversaries can issue false messages to deceive the controllers in other domains [3]. Second, it has less privacy protection for entities on the forwarding path. For example, the public identities of switches are easy to be selected and determined as attack targets (e.g. middle-man attack). Lack of index value protection for a forwarding path can lead to the disclosure of the switches' forwarding order [4], and the path deviation attack [5]. Third, controllers need to collect outcomes of rule enforcement generated by switches in its domain and maintain traceability, which require more storage space [6]. These operations cause heavy load and high communication consumption to controllers

[7]. Execution result of cross-domain forwarding functionality in SDN is a critical factor determining the quality of service (QoS), which motivates us to study multi-controller rule enforcement verification in SDN in this paper.

Recently, some verification schemes in SDN have been proposed in [8–10]. These schemes mainly focused on the verification algorithms to verify dynamic flow policies and analyze service vulnerabilities. Every change of forwarding behaviors will be checked in the real-time verification process, which increases network computing overhead. Many cryptographic techniques are used in authentication schemes [3, 5, 11, 12], such as message authentication code, hash function, and Merkle hash tree. However, these schemes increase the relative latency of the network and the overhead between switches and controllers. Moreover, they are based on the assumption that the controller is trustworthy, which is not practical in the multi-controller network.

Blockchain technology provides a decentralized and distributed network, in which nodes that do not trust each other can still interact successfully [13, 14]. Many schemes [15–17] have applied blockchain technology into SDN, such as Cochain-SC scheme adopting smart contract technology in multi-domain SDN to resist against DDoS attacks. Although the existing studies in [18–20] resorted to the blockchain technology to record all network events, the design of multi-controller rule enforcement verification models still has some challenges when we combine SDN with cryptography and blockchain technology: (i) how to improve the synergistic effect between centralization in SDN and decentralization on blockchain network to optimize network efficiency and security; (ii) how to design the verification scheme to be more efficient and accurate; and (iii) how to protect the privacy of entities and flows on the forwarding path.

To address the above challenges, we propose a novel secure multi-controller rule enforcement verification (BlockREV) mechanism in SDN to guarantee the correctness of cross-domain forwarding. We first leverage blockchain technology to reduce controllers' load and provide privacy protection for controllers and switches by broadcasting transactions with their addresses rather than real identities. Moreover, we use tags as flows' pseudonyms so that the real information of flows can be hidden. A provably secure aggregate signature scheme is designed by cryptography primitive technology to guarantee the effective verification accuracy of multi-controller rule executions. Furthermore, we present a verification algorithm to ensure the correct index of forwarding nodes in each domain through checking the address sequence of switches. The nested hash method of forward paths is used to verify the correct order of operations among domains.

The main contributions are summarized below.

(i) We propose a novel blockchain-enabled multi-controller rule enforcement verification mechanism in SDN, named BlockREV. We adopt the consortium blockchain technology to provide traceability and privacy protection for forwarding operations and solve the problem of mistrust among controllers.

(ii) We provide an address-based aggregate signature scheme on cryptography primitives which is provably secure in the random oracle model. Only designated verifiers can verify signatures signed by switches on the path, so as to provide protection of entities' privacy and index privacy.

(iii) We present a verification algorithm based on forwarding path hash values to check the consistency of forwarding order by means of comparing index values of switches in actual path with that in the configured path.

(iv) The security analysis shows that BlockREV is secure and can resist many kinds of attacks. Performance evaluation demonstrates that our proposed scheme is effective and suitable for multi-controller scenarios in SDN.

The rest of this paper is organized as follows. Section 2 presents some related works on rrule enforcement verification schemes and aggregate signature algorithm. In Section 3, we describe the problem statement. The construction of BlockREV is shown in Section 4, and security analysis is given in Section 5. Section 6 provides the performance evaluation and result analysis. Section 7 concludes this paper.

2. Related Work

In this section, we review the related works, which can be summarized into the following three aspects: rule enforcement verification schemes in SDN, rule enforcement verification schemes on blockchain, and applications of aggregate signature algorithm.

2.1. Rule Enforcement Verification Schemes in SDN. Rule enforcement verification process is an important stage to confirm on the correctness of the rule execution and guarantee the quality of the forwarding service in SDN. Accordingly, there are many works on the rule enforcement verification. A forwarding path verification mechanism was proposed to flag forwarding path by the controller with a custom probe packet [12]. In [10], a validation scheme, named TrustTopo, was used to analyze service vulnerabilities. Furthermore, it adopted the technology of asynchronous rollback to verify a host location and the chaotic model for link verification. In [9], the layer design between controller and network devices, called VeriFlow, achieved the real-time verification of potential violation of key network invariants with a novel incremental algorithm to verify dynamic flow policies. In [3], a multi-controller architecture was proposed with distributed rule store for SDNs with Merkle hash tree to detect rule modifications. However, these schemes increase the relative latency and overhead of controllers. In particular, checking each change of forwarding behaviors in real time results in a large amount of network computation overhead.

2.2. Rule Enforcement Verification Schemes on Blockchain. Blockchain is a decentralized and distributed peer-to-peer network. There are some research studies which integrate blockchain technology and SDN. In [7], the authors introduced a distributed cloud architecture based on blockchain and SDN to provide secure and low-cost computing infrastructure in IoT, in which end-to-end delay is minimized between computing resources and IoT devices. In [21], a new authentication approach was proposed using blockchain and SDN techniques to remove unnecessary reauthentications in repeated handover which is appropriate for 5G network among heterogeneous cells. In [22], a new model, called SDIoBoT, was presented leveraging SDN architecture and the blockchain technology in 5G network, in which an elliptic curve digital signature algorithm was designed to ensure non-repudiation and integrity of the communications in the model. In [17], a blockchain-enabled architecture of controllers was proposed with an efficient authentication method to eliminate the overheads of the traditional blockchain, in which the cluster structure with a new routing protocol was designed to optimize energy consumption and enhance security in IoT. However, these existing research studies do not fully consider energy consumption issues; furthermore, the security proof of cryptographic primitive algorithms needs to be supplemented.

2.3. Applications of Aggregate Signature Algorithm. An aggregate signature scheme compacts a signature set into one short signature to reduce the overhead of verifiers. In [23], an aggregate signature scheme based on bilinear maps was proposed firstly to reduce the size of certificate chains and message size. In the literatures on the improvements of aggregate signature schemes, there are two types: certificate-based schemes [24–26] and certificateless schemes [27–30]. In [31], the authors proposed a certificate-based sequential aggregate signature scheme, in which each node aggregates the previous node's signature and its own information to obtain a new signature. Aggregate signature is aggregated by signers in the specified order, and the order of signers is the key factor for validation. In [32], an aggregate signature scheme based on bilinear pairings with only one designated verifier was proposed to provide privacy protection for signers, which is a certificateless scheme that does not need certificate storage and public key verification. However, sequential aggregate signature scheme is not suitable for the scenario with multiple controllers because it will expose the information of all the previous forwarding nodes and the index value of the paths. Furthermore, we need more designated verifiers in the process of cross-domain forwarding.

3. Problem Statement

In this section, we outline the system model, adversary model, and security assumption. For convenience, some important notations used in the paper are summarized in Table 1.

Table 1: List of notations.

Notation	Definition
$\mathbb{G}_A$	Cyclic additive group
$\mathbb{G}_M$	Cyclic multiplicative group
q	Prime order of $\mathbb{G}_A$ and $\mathbb{G}_M$
P	Generator of $\mathbb{G}_A$
e	Bilinear pairing: $\mathbb{G}_A \times \mathbb{G}_A \longrightarrow \mathbb{G}_M$
$H_1(\cdot), H_2(\cdot)$	Hash functions: $\{0,1\}^* \longrightarrow \mathbb{G}_A$
H_3	Hash function: $\{0,1\}^* \longrightarrow \mathbb{Z}_q^*$
H_4	Hash function: $\{0,1\}^* \longrightarrow \{0,1\}^*$
x	Master key of the system
Y	Public key of the system
i	Domain serial number
j	Index value in the domain Dom_i
ID_{ij}	Identity of the switch s_{ij}
T_{ij}	Registration timestamp of the switch s_{ij}
B_{ij}	Partial private key of the switch s_{ij}
U_{ij}	Public key of the switch s_{ij}
Addr_{ij}	Address of the switch s_{ij}
Tag_f	tag value of flow f
Path_i	Forwarding rules in the domain Dom_i
tx_{ij}	Forwarding transaction published by the switch s_{ij}
outputAddr_{ij}	Output address in tx_{ij}
σ_{ij}	Individual signature generated by the switch s_{ij}
$\widetilde{\sigma}_i$	Aggregate signature generated by the controller c_i
	Message concatenation operation

3.1. System Model. Traditional cross-domain verification solutions may result in single-point failure, high-level overhead, and high maintenance cost. Our new verification model has the following characteristics: unforgeability (compromised entities cannot cheat others by means of blockchain technology), anonymity (entities publish transactions using pseudonymous address), security (the scheme is robust for various attacks), and efficiency (compared with related verification methods, it is more effective).

In this paper, our proposed verification mechanism is inspired by Bitcoin system [33] and the aggregate signature scheme with only one specified verifier [32]. We consider that basic multi-controller architecture is flat, in which each domain decorates only one controller and some switches for its local network. Controllers in different domains intercommunicate through east-west interface, and they are managed by administrator which serves as the master controller. Our proposed decentralized verification system model consists of four parts which are divided into application plane, block-controller plane, data plane, and management layer, as illustrated in Figure 1. Participants involved in the model will be elaborated as follows.

(i) Switches: switches in the data plane are controlled only by the controller in its domain. They are lightweight nodes with finite computing power which are responsible for forwarding flows and publishing forwarding transactions with their signatures on the blockchain. Switches are semitrusted

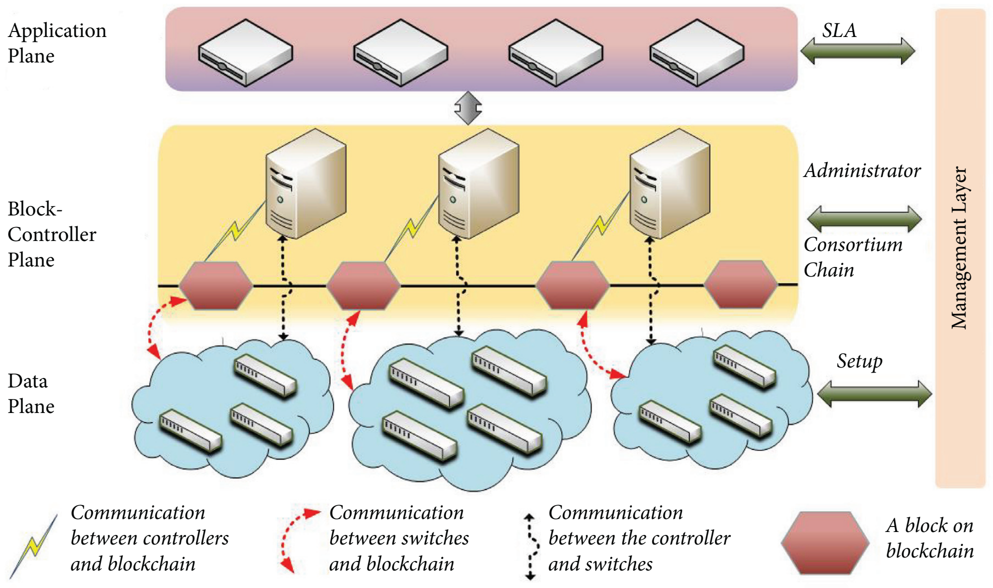

Application Plane
SLA
Block-Controller Plane
Administrator
Consortium Chain
Management Layer
Data Plane
Setup
Communication between controllers and blockchain
Communication between switches and blockchain
Communication between the controller and switches
A block on blockchain

Figure 1: The architecture of BlockREV.

nodes, which may be compromised by adversaries to do malicious behaviors.

(ii) Controllers: distributed controllers in block-controller plane undertake the function of maintaining the local network in their own domains. They are the core of the architecture linking applications and network hardware equipment. Every controller manages switches by means of standard southbound APIs (e.g., OpenFlow [34]) and communicates with application layer through standard northbound APIs. They act as simplified payment verification (SPV) notes on the blockchain just like that in the Bitcoin network and validate transactions published by switches in their own domains without caring for that in others.

(iii) Administrator: administrator node in the management layer is the manager of the block-control plane. It can obtain global information of multi-controller network and customize cross-domain forwarding rules for flows. Administrator node also provides a service of registration for entities and flows in the blockchain network. In addition, it is responsible for verifying the correctness of cross-domain forwarding behaviors, which is the key factor to guarantee the QoS of multi-controller network.

(iv) Verichain: Verichain is a consortium blockchain network, which stores forwarding transaction information and verifies the correctness of the enforcement of forwarding rules. It has the following functions. Rule execution message storage—Verichain stores rule enforcement messages broadcasted by entity notes. It is responsible for recording and forwarding transactions generated by switches and confirming forwarding transactions generated by controllers and the administrator. Rule execution verification—administrator and controllers are qualified to verify the correctness of rule execution results on the basis of distributed ledgers. A special design of signature algorithm in this paper ensures the validity of the verification authority. Ledger update—Verichain expands its distributed ledger with blocks and reaches consensus to record the network status of SDN in real time. By updating the ledger to record the operation behaviors of entities, the execution of flow rules in the network can be monitored.

All entities in SDN are the nodes on Verichain. The administrator is a full node and keeps the whole blockchain information. Although controllers and switches are lightweight nodes, their functions are different. Controllers validate forwarding behavior information of switches in its domain, while switches only publish forwarding transactions and are not responsible for the validation.

3.2. Adversary Model

3.2. Adversary Model. The adversary $\mathscr{A}$ aims to destroy rule execution behaviors and pass the verification with attacks as follows.

Type 1 (*Single-Point Failure Attack*). The controller is a key component of the centralized domain, the compromise of which will cause serious interference in forwarding activities and reduce the availability and reliability of services.

Type 2 (*Middle-Man Attack*). Anonymity easily leads to a middle-man attack in which the adversary may pretend to be a legitimate controller or switch to perform malicious operations.

Type 3 (*Path Deviation Attack*). Malicious behaviors of attackers make the flows unable to be forwarded according to forwarding rules in many ways, e.g. switch bypass, path detour, and out-of-order traversal [11].

In this paper, we make the following assumptions:

(i) The communication channels among entities are secure, and hash functions are one-way and collision resistant.

(ii) Administrator is an honest node managing the entire network. Controllers and switches are relatively credible, which might be compromised in a minority. Additionally, no collusion exists between controllers and switchers.

(iii) Adversary $\mathscr{A}$ cannot obtain the master key of the system but can replace the addresses of legal switches.

4. Construction of BlockREV

In this section, we present the construction of BlockREV mechanism. We first briefly overview the designed mechanism. Then, we present system initialization stages and forwarding process in BlockREV, respectively. At last, based on the nested hash value method and aggregate signature theory, we design the verification scheme of cross-domain forwarding.

4.1. Overview. Controllers and switches register through administrator to gain authority for the admittance of consortium blockchain network. They obtain their addresses and partial private keys in the registration process. These entities communicate anonymously with each other using their addresses without revealing their real identity information on Verichain. This anonymous approach provides better privacy protection. After the system initialization phase is completed, flows are forwarded by the relevant switches in the domain. When the destination node receives these flows, it sends a request to the controller for forwarding out of the domain. Then, the controller confirms whether forwarding rules of flows have been executed correctly in its domain; if so, it submits the cross-domain forwarding request to the controller in the subsequent domain. After two controllers communicate with each other, they issue forwarding instructions to the related switches, and a new round of forwarding by switches begins in the subsequent domain. In order to ensure the correctness of forwarding behaviors, the controllers and the administrator are responsible for verifying the forwarding behaviors in the intra-domain and the cross-domain, respectively.

In our system, we suppose that flows will be forwarded between two domains Dom_1 and Dom_2. Forwarding among more domains can be inferred in the same way. When the flow f proposes a cross-domain forwarding requirement, Admin customizes forwarding rules for f which will be sent to relative controllers and switches. Verichain network stores every forwarding message in the form of transaction, and the correct forwarding actions are guaranteed by verification schemes with cryptography technology.

4.2. System Initialization. The system initialization process in BlockREV comprises three stages: system parameter setting, entity registration, and flow registration.

(i) *System Parameter Setting.* The administrator node Admin generates system parameters. Let additive group $\mathbb{G}_A$ and multiplicative group $\mathbb{G}_M$ be two cyclic groups with the prime order q, and $\widehat{P}$ is the generator of $\mathbb{G}_A$. The bilinear pairing is $e\colon \mathbb{G}_A \times \mathbb{G}_A \longrightarrow \mathbb{G}_M$. The collision-resistant cryptographic hash functions are $H_1, H_2\colon \{0,1\}^* \longrightarrow \mathbb{G}_A$, $H_3\colon \{0,1\}^* \longrightarrow \mathbb{Z}_q^*$, and $H_4\colon \{0,1\}^* \longrightarrow \{0,1\}^*$. Admin chooses a random number $x \in \mathbb{Z}_q^*$, and let $Y = x\widehat{P}$ be the public key and x denote the master key. The system parameter is $\text{Params} = \langle \mathbb{G}_A, \mathbb{G}_M, \widehat{P}, q, e, Y, H_1, H_2, H_3, H_4 \rangle$.

(ii) *Entity Registration.* The SDN structure in this paper includes multiple domains, and the i-th domain Dom_i includes one controller c_i and several switches s_{ij}, where i is the domain serial number and j is the index of the switch in the i-th domain. Entities including controllers and switches register on Verichain through administrator node Admin. They are configured with unique partial private key for signing. Specific steps of registration of a switch are shown in Algorithm 1, in which the function $\text{Addr}(\cdot)$ is a wallet address generation algorithm which is the same used in Bitcoin network. After that, Admin sends the information $(ID_{ij}, U_{ij}, T_{ij}, i, j, \text{Addr}_{ij}, B_{ij})$ to c_i, where ID_{ij} is the identity of switch s_{ij}, U_{ij} is the public key of s_{ij}, T_{ij} is the registration timestamp of s_{ij}, Addr_{ij} is the address of s_{ij} on Verichain, and B_{ij} is the partial private key of s_{ij} generated by Admin. Then, the controller c_i stores and sends this information to switch s_{ij}. The registration process of c_i is similar to that of switches, except for replacing ij with $i0$, and Admin saves $(ID_{i0}, U_{i0}, T_{i0}, i, 0, \text{Addr}_{i0}, B_{i0})$ into its entity registration list ERList, where ID_{i0} is the identity of c_i, U_{i0} is the public key of c_i, T_{i0} is the registration timestamp of c_i, Addr_{i0} is the address of c_i on Verichain, and B_{i0} is the partial private key of c_i generated by Admin.

(iii) *Flow Registration.* There are two types of flows, single flow and multiflow, which will be configured cross-domain forwarding rules by the administrator node after registering on Verichain. Suppose that source host sends the flow f to switch s_{11} in domain Dom_1. Switch s_{11} searches its forwarding list and does not find the flow f's forwarding rules; then, it sends a cross-domain forwarding requirement to controller c_1. After controller c_1 has received this requirement, it sends the information vector $V_f = (\text{header}_f, \text{Addr}_{10}, \text{Addr}_{11}, T_f)$, to the administrator node Admin, where T_f is the registration timestamp of f. After accepting V_f, administrator node Admin computes the hash value $\beta_f = H_1(V_f)$ and gets the tag value $\text{Tag}_f = x\beta_f$ and then saves (V_f, Tag_f) in its flow registration list. According to the global network status, administrator node Admin formulates cross-domain forwarding rules with an ordered set of addresses $\mathrm{Path}_f = \cup \mathrm{Path}_i$, where Path_i denotes the forwarding rule with an ordered set of switches' addresses on the forwarding path in Dom_i.

We assume that the forwarding rules are $s_{11} - s_{12} - \cdots - s_{1m}$ in Dom_1 and $s_{21} - s_{22} - \cdots - s_{2n}$ in Dom_2, and the corresponding set of addresses is $\mathrm{Path}_f = \mathrm{Path}_1 \cup \mathrm{Path}_2$, where $\mathrm{Path}_1 = \{\mathrm{Addr}_{11}, \ldots, \mathrm{Addr}_{1m}\}$ and $\mathrm{Path}_2 = \{\mathrm{Addr}_{21}, \ldots, \mathrm{Addr}_{2n}\}$, $m, n \in \mathbb{Z}^+$. s_{11} and s_{21} are source switches, and s_{1m} and s_{2n} are destination switches in their domains. Administrator node Admin computes $h^1_{\mathrm{path}} = H_4(\mathrm{Path}_1)$, $h^2_{\mathrm{path}} = H_4(\mathrm{Path}_2)$, and $h^f_{\mathrm{path}} = H_4(h^2_{\mathrm{path}}, H_4(h^1_{\mathrm{path}}))$. It updates its forwarding list by $\{V_f, \mathrm{Tag}_f, \mathrm{Path}_f, h^1_{\mathrm{path}}, h^2_{\mathrm{path}}, h^f_{\mathrm{path}}\}$. Finally, administrator node Admin sends $\{\mathrm{Tag}_f, \mathrm{Path}_1, h^1_{\mathrm{path}}\}$ to c_1, and $\{\mathrm{Tag}_f, \mathrm{Path}_2, h^2_{\mathrm{path}}\}$ to c_2, respectively. Controllers c_1 and c_2 then install flow rules at related switches in their own domains through a standard control channel.

When there are l flows to be forwarded with the same forwarding rules at the same time, administrator node Admin will compress their tags into a single node $\mathrm{Tag}_{\tilde{f}} = \sum_{i=1}^{l} \mathrm{Tag}_f$ in the registration process, and then multiple flows will be forwarded in batch processing.

4.3. Forwarding Process in BlockREV.

4.3. Forwarding Process in BlockREV. As shown in Figure 2, the forwarding process in BlockREV includes three phases: intra-domain forwarding, cross-domain forwarding, and forwarding to destination host. Specific phases are described as follows.

4.3.1. Intra-Domain Forwarding. In this phase, the flow f will be forwarded from source switch s_{11} to destination switch s_{1m} in domain Dom_1 or from source switch s_{21} to destination switch s_{2n} in domain Dom_2. After forwarding rule configuration of the flow f by the controller, the related switches start forwarding. After completing the forwarding, switch s_{ij} generates a forwarding transaction tx_{ij} with its signature and publishes it on Verichain. An illustration of block data structure and transaction information is shown in Figure 3. The specific verification algorithm of forwarding order and aggregate signature scheme will be described in Section 4.4. When the flow f reaches switch s_{1m}, cross-domain forwarding phase will start. After the flow f reaches switch s_{2n}, it will be forwarded to the destination host in the last phase.

4.3.2. Cross-Domain Forwarding. In this phase, the flow f will be forwarded in cross-domain manner from switch s_{1m} to switch s_{21} between domain Dom_1 and domain Dom_2. When the flow f arrives at switch s_{1m} in domain Dom_1, switch s_{1m} sends an out-domain forwarding request to controller c_1. As a domain manager, controller c_1 checks whether the flow f has been forwarded correctly in its domain or not. It firstly verifies whether the order of switches in Path_1 is correct in Dom_1 or not. Then, controller c_1 checks the validity of signatures signed in forwarding transactions by switches on the forwarding path. The specific validation process will be explained later. If the verification is passed, controller c_1 aggregates m signatures made by switches in Path_1 into one signature and sends it to administrator node Admin for the whole path verification. The specific verification scheme will be described later. Controller c_1 publishes a confirmed forwarding transaction tx_{10} on Verichain; after that, c_1 sends the cross-domain forwarding request of the flow f to controller c_2.

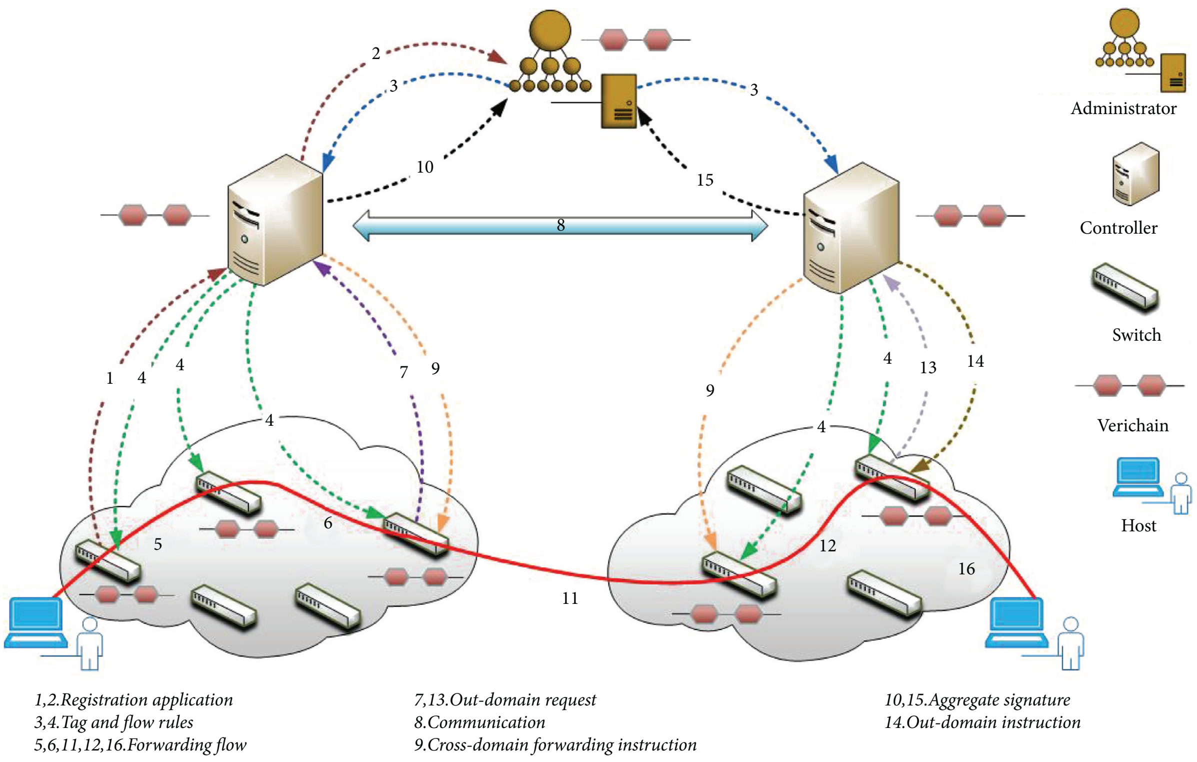
When controller c_2 receives the request from controller c_1, it searches its forwarding list to find forwarding rules of Tag_f. If it makes, controller c_2 checks transactions published by controller c_1 on Verichain to confirm whether forwarding behaviors in domain Dom_1 have been verified by controller c_1. If the confirm information in tx_{10} is Yes, controller c_2 gives an affirmative response to controller c_1 with the address information of source note s_{21} and dispenses the forwarding instruction to s_{21}. Otherwise, c_2 rejects and communicates the situation to Admin. If c_1 receives the affirmative response from c_2, it responds to s_{1m} that f should be forwarded to s_{21}. Then, s_{21} receives f and begins a new round of forwarding. The specific cross-domain communication process is shown in Figure 4.

4.3.3. Forwarding to Destination Host. In this phase, f will be forwarded from s_{2n} to the destination host. When f reaches s_{2n}, s_{2n} sends an out-domain forwarding request to c_2. As a domain manager, c_2 verifies rule enforcement just like c_1 has done. After receiving the aggregate signatures sent by c_1 and c_2, Admin will verify whether the cross-domain forwarding rules have been implemented correctly or not. It first checks the correctness of the forwarding node order and then verifies the validity of two aggregate signatures, respectively, to guarantee the quality of forwarding service. The specific validation process is described later. After that, Admin publishes a confirmed forwarding transaction on Verichain. When the source host or the destination host wants to estimate the correctness of forwarding behaviors in SDN to ensure the security of f in the process of transmission, he may look up the confirmed forwarding transaction about Tag_f published by Admin through any full node on Verichain.

4.4. Validation Process in BlockREV. In this section, we design appropriate forwarding verification schemes for controllers and administrator based on the nested hash value method and aggregate signature theory. The technology of cryptography and blockchain is combined with SDN to ensure the correctness and effectiveness of the verification scheme. We will describe the verification process in two scenarios: intra-domain verification and cross-domain verification.

```
    **Input**: $ID_{ij}$: identity of $s_{ij}$;
    $i$: domain serial number, $i \in \mathbb{Z}^{+}$;
    $j$: index value of the switch in $Do\ m_i$, $j \in \mathbb{Z}^{+}$;
    $x$: master key of the system;
    $T_{ij}$: registration timestamp of $s_{ij}$;
    ERList: entity registration list;
    **Output**: $B_{ij}$: partial private key of $s_{ij}$;
(1) $s_{ij}$ selects a random number $a_{ij} \in \mathbb{RZ}_q^*$ as a secret value and computes its public key
    $U_{ij} \leftarrow a_{ij}\widehat{P}$ and address $\mathrm{Addr}_{ij} \leftarrow \mathrm{Addr}(U_{ij})$.
(2) $s_{ij}$ sends $(ID_{ij}, U_{ij}, T_{ij}, i, j, \mathrm{Addr}_{ij})$ to $A\ dm\ in$;
(3) Admin checks
(4) **if** $ID_{ij} \in$ ERList or $U_{ij} \in$ ERList or $\mathrm{Addr}_{ij} \in$ ERList **then**
(5) return fails;
(6) **else**
(7) Admin computes: $D_{ij} \leftarrow H_1(\mathrm{Addr}_{ij})$; $B_{ij} \leftarrow xD_{ij}$; Admin saves $(ID_{ij}, U_{ij}, T_{ij}, i, j, \mathrm{Addr}_{ij}, B_{ij})$ into ERList.
(8) **end if**
(9) return $B_{ij}$;
```

Algorithm 1: Registration of a switch.



2
3
3
10
15
8
1
4
4
4
7
9
9
4
4
13
14
5
6
11
12
16
Administrator
Controller
Switch
Verichain
Host
1,2.Registration application
3,4.Tag and flow rules
5,6,11,12,16.Forwarding flow
7,13.Out-domain request
8.Communication
9.Cross-domain forwarding instruction
10,15.Aggregate signature
14.Out-domain instruction

Figure 2: Forwarding process of BlockREV.

4.4.1. Intra-Domain Validation. In the intra-domain forwarding phase, controller c_i traces forwarding transactions about Tag_f on Verichain, which have been published by switches in its domain. First, c_i verifies whether the forwarding order of switches is consistent with that in its memory. It calculates the hash value of the public key in the subsequent transaction in order to compare the result with the output address in the current transaction. If they are equal, it concatenates the output address into Path_i. After making sure that the order of switches in all adjacent transactions is correct, c_i calculates the hash value $H_4(\mathrm{Path}_i)$. By comparing it with the hash value h^i_{path}, c_i can confirm whether the forwarding order of switches in its domain is correct or not. Specific steps are shown in Algorithm 2. Second, c_i checks individual signatures signed by switches in Dom_i and then aggregates all signatures into a

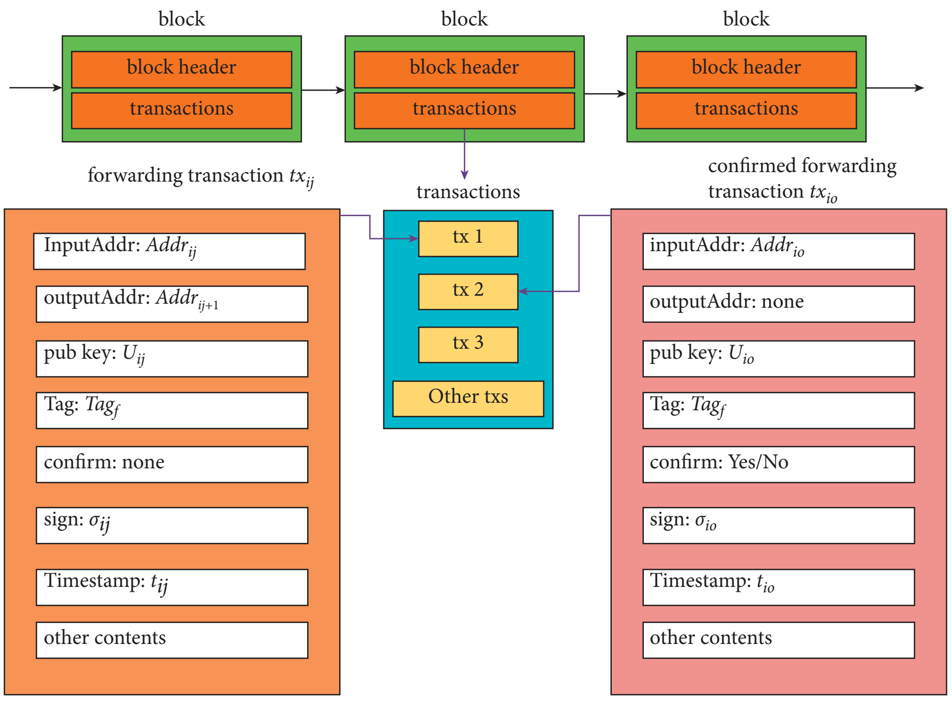


FIGURE 3: The structure of Verichain.

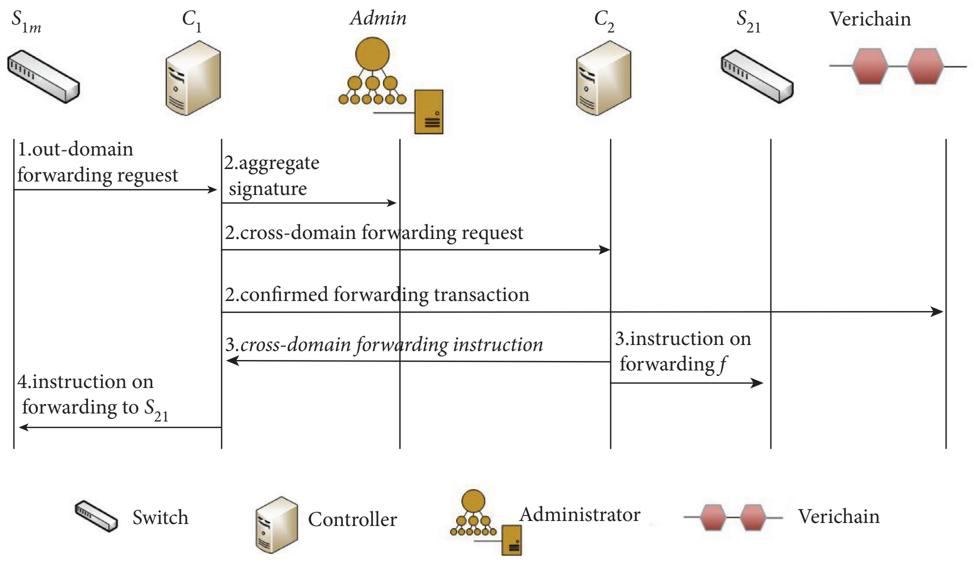

S_{1m}
C_1
Admin
C_2
S_{21}
Verichain
1.out-domain forwarding reguest
2.aggregate signature
2.cross-domain forwarding request
2.confirmed forwarding transaction
3.cross-domain forwarding instruction
3.instruction on forwarding f
4.instruction on forwarding to S_{21}
Switch
Controller
Administrator
Verichain

FIGURE 4: Cross-domain communication process between c_1 and c_2.

single one which will be sent to Admin for cross-domain validation. The specific steps of individual signatures verification will be explained in the aggregation signature scheme later.

4.4.2. Cross-Domain Validation. In the process of forwarding to destination host phase, Admin looks over the information on Verichain and obtains all forwarding transactions associated with Tag_f published by switches in different domains. First, Admin judges whether the cross-domain forwarding order of switches is correct. It verifies the forwarding order in Dom_1 and Dom_2, respectively, and then it verifies whether the equation $\text{outputaddr}_{1m} = \text{Addr}(U_{21})$ holds or not according to the information in tx_{1m} and tx_{21}. If it holds, Admin calculates the nested hash value $H_4(h^2_{\text{path}}, H_4(h^1_{\text{path}}))$. By comparing it with the hash value h^f_{path}, Admin can confirm whether the forwarding order of switches in multiple domains is correct or not. Specific steps are shown in Algorithm 3. Second, Admin verifies aggregate signatures from c_1 and c_2 to effectively ensure the validity of the multi-controller rule enforcement verification.

```
Input: $tx_{ij}$: forwarding transaction published by $s_{ij}$;
$U_{ij}$: public key of $s_{ij}$;
outputAddr$_{ij}$: output address in $tx_{ij}$;
$m, n$: $m, n \in \mathbb{Z}^+$;
$i, k$: $i = 1, k = m$ or $i = 2, k = n$;
$h^i_{\text{path}}$: hash value of Path$_i$;
Output: valid, invalid;
(1) $c_i$ traces back all $tx_{ij}$ about Tag$_f$ published by switches in Dom$_i$;
(2) Path$_i$←Addr$_{i1}$;
(3) for each $j \in [1, k-1]$ do
(4) if Addr$_{i1}$ == Addr$(U_{i1})$ and outputAddr$_{ij}$ == Addr$(U_{ij+1})$ then
(5) $j \leftarrow j+1$;
(6) Path$_i$←Path$_i$‖Addr$_{ij+1}$;
(7) end if
(8) end for
(9) if $h^i_{\text{path}}$ == $H_4$(Path$_i$) then
(10) return valid
(11) else
(12) return invalid
(13) end if
```

Algorithm 2: Verification of the order of switches in Dom$_i$.

Our proposed aggregation signature scheme is defined by six polynomial-time bounded algorithms: Setup, Key-Gen, Individual-Sign, Individual-Verify, Aggregate-Sign, and Aggregate-Verify. Specific steps are described as follows:

(i) *Setup*. Admin inputs the security parameter λ and outputs the system parameter Params $= \langle \mathbb{G}_A, \mathbb{G}_M, \widehat{P}, q, e, Y, H_1, H_2, H_3, H_4 \rangle$, where $Y = x\widehat{P}$ is the public key and $x \in_R \mathbb{Z}_q^*$ is the master key.

(ii) *Key-Gen*. In the entity registration process, s_{ij} obtains its partial secret key B_{ij}, and c_i obtains its partial secret key B_{i0}.

(iii) *Individual-Sign*. For tx_{ij}, let $m_{ij} = \{U_{ij}, \text{Tag}_f, T^f_{ij}\}$. s_{ij} chooses a random number $r_{ij} \in \mathbb{Z}_q^*$ and computes $R_{ij} = r_{ij}\widehat{P}$, $H_{ij} = H_2(a_{ij}U_{i0}Y)$, $h_{ij} = H_3(m_{ij}, U_{ij})$, and $W_{ij} = (r_{ij} + a_{ij})H_{ij} + h_{ij}B_{ij}$. Then s_{ij}'s individual signature is $\sigma_{ij} = (R_{ij}, W_{ij})$.

(iv) *Individual-Verify*. c_i is responsible for verifying signatures in its domain. For k signatures $\sigma_{ij} = (R_{ij}, W_{ij})$, $1 \le j \le k$, $k \in \{m, n\}$, c_i performs the following steps:

(a) Tracing on Verichain for all operating transactions associated with Tag$_f$ in Dom$_i$ and judging whether the order of addresses is the same with that in Path$_i$, as shown in Algorithm 2.

(b) Verifying the signature of s_{ij} on Path$_i$. c_i calculates $H_{ij} = H_2(a_{i0}U_{ij}Y)$, $h_{ij} = H_3(m_{ij}, U_{ij})$, and $D_{ij} = H_1(\text{Addr}_{ij})$. Checking whether the equation

$$e\left(W_{ij}, \widehat{P}\right) = e\left(h_{ij}D_{ij}, Y\right) \cdot e\left(H_{ij}, R_{ij}\right) \cdot e\left(H_{ij}, U_{ij}\right) \tag{1}$$

holds or not. If it holds, c_i accepts σ_{ij}.

(v) *Aggregate-Sign*. For k accepted individual signatures $\sigma_{ij} = (R_{ij}, W_{ij})$ given by distinct switches on *Path$_i$*, where $i = 1, k = m$ or $i = 2, k = n$, c_i calculates $\widetilde{W}_i = \sum_{j=1}^k W_{ij}$ and generates the aggregate signature $\widetilde{\sigma}_i = (R_{i1}, R_{i2}, \ldots, R_{ik}, \widetilde{W}_i)$.

(vi) *Aggregate-Verify*. *A dm in* completes the verification of aggregate signatures which are generated by c_1 and c_2, respectively. When *A dm in* receives $\widetilde{\sigma}_i = (R_{i1}, R_{i2}, \ldots, R_{ik}, \widetilde{W}_i)$, it performs the following steps:

(a) Tracing all the operating transactions associated with Tag$_f$ on Verichain and judging whether the order of transactions is the same with Path$_f$, in order to ensure that the sequence of forwarding nodes is correct. The trace process is similar to that in Individual-Verify step, except for the extra validations pubkeyhash$_{1m}$ = Addr(U_{21}) and $h^f_{\text{path}} = H_4(h^2_{\text{path}}, H_4(h^1_{\text{path}}))$ needed to be done, as shown in Algorithm 3.

(b) Computing $H_{ij} = H_2(xU_{ij}U_{i0})$, $h_{ij} = H_3(m_{ij}, U_{ij})$ and $D_{ij} = H_1(\text{Addr}_{ij})$ and checking whether the equation

$$e\left(\widetilde{W}_i, \widehat{P}\right) = e\left(\sum_{j=1}^k h_{ij}D_{ij}, Y\right) \cdot \prod_{j=1}^k e\left(H_{ij}, R_{ij}\right) \prod_{j=1}^k e\left(H_{ij}, U_{ij}\right) \tag{2}$$

holds or not. If it holds, Admin accepts the aggregate signatures $\widetilde{\sigma}_i$.

Because of the properties of bilinear pairings, we can conclude that the aggregate signature scheme mentioned above is correct.

$$\begin{aligned} e\left(\widetilde{W}_i, \widehat{P}\right) &= e\left(\sum_{j=1}^{k} W_{ij}, \widehat{P}\right) = e\left(\sum_{j=1}^{k}\left(r_{ij} + a_{ij}\right)H_{ij} + h_{ij}B_{ij}, \widehat{P}\right) \\ &= e\left(\sum_{j=1}^{k} xh_{ij}D_{ij}, \widehat{P}\right) \cdot \prod_{j=1}^{k} e\left(r_{ij}H_{ij}, \widehat{P}\right) \cdot \prod_{j=1}^{k} e\left(a_{ij}H_{ij}, \widehat{P}\right) \\ &= e\left(\sum_{j=1}^{k} h_{ij}D_{ij}, Y\right) \cdot \prod_{j=1}^{k} e\left(H_{ij}, R_{ij}\right) \cdot \prod_{j=1}^{k} e\left(H_{ij}, U_{ij}\right). \end{aligned} \tag{3}$$

If multiple flows are assigned with the same forwarding rules simultaneously, the batch verification technique is adopted to improve the efficiency of validation operations in our mechanism. The tag of batch flows is $\mathrm{Tag}_{\widetilde{f}}$, which will replace Tag_f in the above scheme to perform batch verification.

5. Security Analysis

We adopt the adaptive chosen-message security model to prove our algorithm. In this model, given a designated address, the adversary $\mathscr{A}$ wants to obtain the existential forgery of an aggregate signature.

Definition 1. Bilinear map: let $\mathbb{G}_A$ and $\mathbb{G}_M$ be an additive cyclic group and a multiplicative cyclic group, respectively, with the same prime order q, and $\widehat{P}$ is a generator of $\mathbb{G}_A$. Let e be a bilinear map such that e: $\mathbb{G}_A \times \mathbb{G}_A \longrightarrow \mathbb{G}_M$ which has the following properties:

(1) Bilinear: for all $\widehat{P}_1, \widehat{P}_2 \in \mathbb{G}_A$ and $a, b \in \mathbb{RZ}_q^*$, $e(a\widehat{P}_1, b\widehat{P}_2) = e(\widehat{P}_1, \widehat{P}_2)^{ab}$

(2) Non-degenerate: $e(\widehat{P}, \widehat{P}) \neq 1$, where $\widehat{P}$ is the generator of $\mathbb{G}_A$.

(3) Computable: for all $\widehat{P}_1, \widehat{P}_2 \in \mathbb{G}_A$, $e(\widehat{P}_1, \widehat{P}_2)$ is efficiently computable.

Definition 2. Computational Diffie–Hellman (CDH) problem: Given $\widehat{P}, a\widehat{P}, b\widehat{P} \in \mathbb{G}_A$ as input, compute $ab\widehat{P} \in \mathbb{G}_A$ for unknown $a, b \in \mathbb{RZ}_q^*$.

We say that CDH problem is (t, ε)-hard if there is no algorithm that can solve the problem with probability no more than ε in time at most t.

Definition 3. An aggregate signature scheme $\widehat{\Pi}$ is existentially unforgeable under an adaptive chosen-message model, if for all probabilistic polynomial-time adversaries $\mathscr{A}$, there is a negligible function $\widehat{N}$ for the advantage $\mathrm{Adv}_{\mathscr{A}}$ such that

$$\mathrm{Adv}_{\mathscr{A}} = \Pr\left[\mathrm{AggSigForge}_{\mathscr{A},\widehat{\Pi}} = 1\right] \leq \widehat{N}. \tag{4}$$

$\mathscr{A}$ wins if his advantage, $\mathrm{Adv}_{\mathscr{A}}$, is non-negligible. His aggregate signature is valid and non-trivial, i.e., $\mathscr{A}$ does not inquire about the signatures of at least one message under the designated address.

Definition 4. A forger $\mathscr{A}$ is $(t, \varepsilon, q_{H_1}, q_{H_2}, q_{H_3}, q_{psk}, q_{fsk}, \backslash\backslash q_{ar}, q_s, k)$ – breaks an aggregate signature scheme in BlockREV with the adaptive chosen-message model if: $\mathscr{A}$ makes at most q_{H_1}, q_{H_2} and q_{H_3} queries to the hash function, at most q_{psk} queries to the partial secret key oracle, at most q_{fsk} queries to the full secret key oracle, at most q_{ar} queries to the address replacement oracle, at most q_s queries to the signing oracle, runs in time at most t with at most k users, and $\mathrm{Adv}_{\mathscr{A}}$ is at least ε.

Definition 5. An aggregate signature scheme in BlockREV is $(t, \varepsilon, q_{H_1}, q_{H_2}, q_{H_3}, q_{\mathrm{psk}}, q_{\mathrm{fsk}}, q_{ar}, q_s, k)$ – secure against existential forgery if there is no adversary $(t, \varepsilon, q_{H_1}, q_{H_2}, q_{H_3}, q_{\mathrm{psk}}, q_{\mathrm{fsk}}, q_{ar}, q_s, k)$ – breaks it in the adaptive chosen-message model.

Theorem 1. *If there exists a forger $\mathscr{A}$ that can $(t, \varepsilon, q_{H_1}, q_{H_2}, q_{H_3}, \backslash\backslash q_{psk}, q_{fsk}, q_{ar}, q_s, k)$ – break the aggregate signature scheme in BlockREV by non-negligible probability ε, an algorithm $\mathscr{C}$ can solve an instance of CDH problem in polynomial time $t' \leq t + (q_{H_1} + q_{H_2} + q_{psk} + 3q_s + 3k - 1)t_{sm} + t_{inv}$ with non-negligible probability $\varepsilon\prime \geq \varepsilon/q_{H_1} \cdot e^{(1-k-q_s-q_{fsk})/q_{H_1}}$. Here t_{sm} is the time of a scalar multiplication in $\mathbb{G}_A$, t_{inv} is the time to compute an inverse in $\mathbb{Z}_q$, and e is the natural logarithm base.*

Proof. Algorithm $\mathscr{C}$ simulates a challenger. Assume that $\mathscr{C}$ is given an CDH problem instance $(q, \widehat{P}, a\widehat{P}, b\widehat{P})$ and will interact with $\mathscr{A}$ as follows to compute $ab\widehat{P}$. Let Addr^* be the target victim, and the detailed process is as follows.

Setup: $\mathscr{C}$ sets the system parameters—Params = $\langle \mathbb{G}_A, \mathbb{G}_M, \widehat{P}, q, e, Y, H_1, H_2, H_3 \rangle$, public key $Y = a\widehat{P}$. H_1, H_2, H_3 are three random oracles controlled by $\mathscr{C}$. Then, $\mathscr{C}$ sends *Params* to $\mathscr{A}$.

Queries: $\mathscr{C}$ maintains empty list $\mathscr{L}_{H_1}$, $\mathscr{L}_{H_2}$, $\mathscr{L}_{H_3}$, $\mathscr{L}_{\mathrm{psk}}$, $\mathscr{L}_{\mathrm{fsk}}$, $\mathscr{L}_{ar}$, $\mathscr{L}_s$, and $\mathscr{A}$ simulates the oracle queries in the following types.

(i) H_1 – query: upon receiving the entry Addr_{ij}, $\mathscr{C}$ performs the following:

  (a) $\mathscr{C}$ checks whether existing $(\mathrm{Addr}_{ij}, \mathrm{coin}_{ij}, \theta_{ij}, D_{ij})\backslash\backslash \in \mathscr{L}_{H_1}$ or not. If so, $\mathscr{C}$ sends D_{ij} to $\mathscr{A}$.

  (b) Otherwise, $\mathscr{C}$ generates a random coin $\mathrm{coin}_{ij} \in \{0, 1\}$ so that $\Pr[coin_{ij} = 0] = 1/q_{H_1}$. $\mathscr{C}$ randomly selects $\theta_{ij} \in \mathbb{Z}_q^*$. If $\mathrm{coin}_{ij} = 0$ holds, $\mathscr{C}$ computes $D_{ij} = \theta_{ij}b\widehat{P}$. If $\mathrm{coin}_{ij} = 1$ holds, $\mathscr{C}$ computes $D_{ij} = \theta_{ij}\widehat{P}$. $\mathscr{C}$ responds to $\mathscr{A}$ with D_{ij} and saves $(\mathrm{Addr}_{ij}, \mathrm{coin}_{ij}, \theta_{ij}, D_{ij})$ into the hash list $\mathscr{L}_{H_1}$.

(ii) H_2 – query: upon receiving the entry (U_{ij}, U_{i0}), $\mathscr{C}$ checks whether existing $(U_{ij}, U_{i0}, \eta_{ij}, H_{ij}) \in \mathscr{L}_{H_2}$ or not. If so, $\mathscr{C}$ sends H_{ij} to $\mathscr{A}$; otherwise, $\mathscr{C}$

```
     Input: $tx_{ij}$: forwarding transaction published by $s_{ij}$;
     $U_{ij}$: public key of $s_{ij}$;
     outputAddr$_{ij}$: output address in $tx_{ij}$;
     $m, n$: $m, n \in \mathbb{Z}^+$;
     $h^i_{\text{path}}$: hash value of Path$_i$;
     $h^f_{\text{path}}$: forwarding path hash value of $f$;
     Output: valid, invalid;
(1)  Admin traces back all $tx_{ij}$ about Tag$_f$;
(2)  $i \leftarrow 1; k \leftarrow m$;
(3)  if the result of executing Algorithm 2 is "valid" then
(4)  $h^1_{\text{path}} \leftarrow H_4(\text{Path}_1)$;
(5)  end if
(6)  $i \leftarrow 2; k \leftarrow n$;
(7)  if the result of executing Algorithm 2 is "valid" then
(8)  $h^2_{\text{path}} \leftarrow H_4(\text{Path}_2)$;
(9)  end if
(10) if outputAddr$_{1m}$ == Addr$(U_{21})$ and $h^f_{\text{path}}$ == $H_4(h^2_{\text{path}}, H_4(h^1_{\text{path}}))$ then
(11) return valid
(12) else
(13) return invalid
(14) end if
```

Algorithm 3: Verification of the cross-domain forwarding order of switches.

randomly selects $\eta_{ij} \in \mathbb{Z}_q^*$, computes $H_{ij} = \eta_{ij}\widehat{P}$, sends H_{ij} to $\mathscr{A}$, and saves $(U_{ij}, U_{i0}, \eta_{ij}, H_{ij})$ into the hash list $\mathscr{L}_{H_2}$.

(iii) H_3 – query: upon receiving the entry (m_{ij}, U_{ij}), $\mathscr{C}$ checks whether existing $(m_{ij}, U_{ij}, h_{ij}) \in \mathscr{L}_{H_3}$ or not. If so, $\mathscr{C}$ sends h_{ij} to $\mathscr{A}$; otherwise, $\mathscr{C}$ randomly selects $h_{ij} \in \mathbb{Z}_q^*$, sends h_{ij} to $\mathscr{A}$, and saves (m_{ij}, U_{ij}, h_{ij}) into the hash list $\mathscr{L}_{H_3}$.

(iv) PartialSecretKey – query: upon receiving the entry Addr$_{ij}$, $\mathscr{C}$ performs the following:

(a) $\mathscr{C}$ checks whether existing $(\text{Addr}_{ij}, \text{coin}_{ij}, B_{ij}) \in \mathscr{L}_{H_{\text{psk}}}$ or not. If so, $\mathscr{C}$ sends B_{ij} to $\mathscr{A}$.

(b) Otherwise, $\mathscr{C}$ makes H_1 – query on Addr$_{ij}$. If coin$_{ij} = 0$ holds, $\mathscr{C}$ computes $B_{ij} = \theta_{ij}bY$. If $coin_{ij} = 1$ holds, $\mathscr{C}$ computes $B_{ij} = \theta_{ij}Y$. $\mathscr{C}$ responses B_{ij} to $\mathscr{A}$ and saves $(\text{Addr}_{ij}, \text{coin}_{ij}, B_{ij})$ into the partial secret key list $\mathscr{L}_{\text{psk}}$.

(v) FullSecretKey – query: upon receiving the address Addr$_{ij}$, $\mathscr{C}$ performs the following:

(a) $\mathscr{C}$ checks whether existing $(\text{Addr}_{ij}, a_{ij}, B_{ij}) \in \mathscr{L}_{fsk}$ or not. If so, $\mathscr{C}$ sends (a_{ij}, B_{ij}) to $\mathscr{A}$.

(b) Otherwise, $\mathscr{C}$ makes H_1 – query on Addr$_{ij}$. If coin$_{ij} = 0$ holds, $\mathscr{C}$ aborts and outputs $\perp$. If coin$_{ij} = 1$ holds, $\mathscr{C}$ randomly selects $a_{ij} \in \mathbb{Z}_q^*$ and makes PartialSecretKey – query on Addr$_{ij}$. Then, $\mathscr{C}$ responses (a_{ij}, B_{ij}) to $\mathscr{A}$ and saves $(\text{Addr}_{ij}, a_{ij}, B_{ij})$ into the full secret key list $\mathscr{L}_{\text{fsk}}$.

(vi) AddressePlacement – query: $\mathscr{A}$ randomly selects $a^*_{ij} \in \mathbb{Z}_q^*$, computes $U^*_{ij} = a^*_{ij}\widehat{P}$ and $A\ dd\ r^*_{ij} = A\ dd\ r(U^*_{ij})$, and sends $(A\ dd\ r^*_{ij}, U^*_{ij})$ to $\mathscr{C}$ for the replacement of any original legitimate address. $\mathscr{C}$ saves $(\text{Addr}^*_{ij}, U^*_{ij})$ into the list $\mathscr{L}_{ar}$.

(vii) Sign – query: upon receiving the entry $(\text{Addr}_{ij}, U_{ij}, U_{i0}, m_{ij})$, $\mathscr{C}$ performs the following:

(a) $\mathscr{C}$ checks whether existing $(\text{Addr}_{ij}, U_{ij}, U_{i0}, m_{ij}, r_{ij}, W_{ij}) \in \mathscr{L}_s$ or not. If so, $\mathscr{C}$ computes $R_{ij} = r_{ij}\widehat{P}$ and sends (R_{ij}, W_{ij}) to $\mathscr{A}$.

(b) Otherwise, $\mathscr{C}$ randomly selects $r_{ij} \in \mathbb{Z}_q^*$ and computes $R_{ij} = r_{ij}\widehat{P}$ and $W_{ij} = (R_{ij} + U_{ij})\eta_{ij} + h_{ij}\theta_{ij}Y$. $\mathscr{C}$ responds to $\mathscr{A}$ with (R_{ij}, W_{ij}) and saves $(\text{Addr}_{ij}, U_{ij}, U_{i0}, m_{ij}, r_{ij}, W_{ij})$ into the sign list $\mathscr{L}_s$.

Forge. After adaptive queries, $\mathscr{A}$ outputs the aggregate signature $\widetilde{\sigma}^*_i = (R^*_{i1}, R^*_{i2}, \ldots, R^*_{ik}, \widetilde{W}^*_i)$ on $\text{Addr}^*_{i1}, \ldots, \text{Addr}^*_{ik}$ and $m^*_{i1}, \ldots, m^*_{ik}$, in which the messages are all distinct, and at least one address $A\ dd\ r^*_{ij}$ does not perform the full secret key query; furthermore, m^*_{ij} does not make sign-query.

If $\text{Addr}^* \notin \{\text{Addr}^*_{i1}, \text{Addr}^*_{i2}, \ldots, \text{Addr}^*_{ik}\}$, $\mathscr{C}$ outputs failure and halts. Otherwise, let $\text{Addr}^* = \text{Addr}^*_{i1}$, and $\mathscr{C}$ proceeds only if $coin_{i1} = 0$ and for $2 \le j \le k$, $coin_{ij} = 1$. Since the aggregate signature $\widetilde{\sigma}^*_i$ is valid, it must satisfy verification equation (2), and we can get

$$e\left(\widetilde{W}_i,\widehat{P}\right)=e\left(W_{i1},\widehat{P}\right)e\left(\sum_{j=2}^{k}W_{ij},\widehat{P}\right)=e\left(\left(r_{i1}+a_{i1}\right)H_{i1}+h_{i1}B_{i1},\widehat{P}\right)\cdot e\left(\sum_{j=2}^{k}\left(R_{ij}+U_{ij}\right)\eta_{ij}+h_{ij}\theta_{ij}Y,\widehat{P}\right)$$
$$=e\left(\eta_{i1}\widehat{P},R_{i1}\right)\cdot e\left(\eta_{i1}\widehat{P},U_{i1}\right)\cdot e\left(h_{i1}\theta_{i1}b\widehat{P},Y\right)\cdot\prod_{j=2}^{k}e\left(\eta_{ij}\widehat{P},R_{ij}\right)\cdot\prod_{j=2}^{k}e\left(\eta_{ij}\widehat{P},U_{ij}\right)\cdot\prod_{j=2}^{k}e\left(h_{ij}\theta_{ij}\widehat{P},Y\right). \tag{5}$$

Then, $\mathscr{C}$ outputs $ab\widehat{P}=(h_{i1}\theta_{i1})^{-1}\cdot(\widetilde{W}_i-\eta_{i1}R_{i1}-\eta_{i1}U_{i1}-\sum_{j=2}^{k}(\eta_{ij}R_{ij}+\eta_{ij}U_{ij}+h_{ij}\theta_{ij}Y))$ as the solution of CDH problem.

Suppose that

(1) Event E_{i1} represents that $\mathscr{C}$ does not output $\perp$ at FullSecretKey – query and Sign – query stages.

(2) Event E_{i2} represents that $\mathscr{A}$ generates a valid aggregate signature forgery.

(3) Event E_{i3} represents that $\mathscr{C}$ does not abort at the forge stage.

The probabilities of solving the CDH problem by $\mathscr{C}$ are as follows:

$$\Pr\left[E_{i1}\right]\geq\left(1-\frac{1}{q_{H_1}}\right)^{q_{fsk}+q_s},\Pr\left[E_{i2}|E_{i1}\right]\geq\varepsilon,\Pr\left[E_{i3}|E_{i1}E_{i2}\right]\geq\frac{1}{q_{H_1}}\left(1-\frac{1}{q_{H_1}}\right)^{k-1}. \tag{6}$$

Therefore,

$$\Pr\left[E_{i1}E_{i2}E_{i3}\right]=\Pr\left[E_{i1}\right]\cdot\Pr\left[E_{i2}|E_{i1}\right]\cdot\Pr\left[E_{i3}|E_{i1}E_{i2}\right]\geq\frac{\varepsilon}{q_{H_1}}\left(1-\frac{1}{q_{H_1}}\right)^{q_{\text{fsk}}+q_s+k-1}. \tag{7}$$

When three events have all happened, algorithm $\mathscr{C}$ can successfully solve the CDH problem with a non-negligible probability

$$\varepsilon'\geq\Pr\left[E_{i1}E_{i2}E_{i3}\right]\geq\frac{\varepsilon}{q_{H_1}}\cdot e^{\left(1-k-q_s-q_{\text{fsk}}\right)/q_{H_1}}. \tag{8}$$

$\mathscr{C}$'s extra time cost includes that for computing at most one scalar multiplication in $\mathbb{G}_A$ on each H_1 – query, H_2 – query, and PartialSecretKey – query and three scalar multiplications on each sign – query. After the forgery operation is completed, the time cost should be plus at most the time for computing $(3k-1)$ scalar multiplication in $\mathbb{G}_A$ and an inverse computation in $\mathbb{Z}_q$. Therefore, $\mathscr{C}$ can solve an instance of CDH problem in polynomial time $t'\leq t+(q_{H_1}+q_{H_2}+q_{\text{psk}}+3q_s+3k-1)t_{sm}+t_{\text{inv}}$.

We have the conclusion that assuming the CDH problem is hard, the proposed aggregate signature scheme in BlockREV is existentially unforgeable against adaptive chosen-message attacks in the oracle model.

We next show security properties of BlockREV mechanism in the context of typical attacks.

(i) Resistance to single-point failure attack: the information on blockchain is transparent and tamper resistant. When a controller is compromised, the administrator can still verify the forwarding behaviors by tracing the transactions on the blockchain and determine the reliability of the controller.

(ii) Resistance to middle-man attack: due to the cryptography theory, transactions published by malicious switches cannot pass signature verification. Because the controller can only verify signatures of switches in its domain, a malicious controller cannot obtain the information about path index and switches' identities in other domains.

(iii) Resistance to path deviation attack: based on public forwarding address information in transactions, the order of forwarding nodes is verified by controllers to ensure the correct execution of forwarding. Through nesting method of forwarding path hash values, the administrator guarantees the correct index of different domains. In addition, our novel aggregate signature scheme provides the security in cryptography technology. □

6. Performance and Evaluation

In this section, we evaluate the performance of our algorithms in terms of computation cost and communication cost compared with other existing schemes.

6.1. Analysis of Computation Cost. We compare the performance of the aggregate signature scheme in BlockREV with two related aggregate signature schemes [27, 35] and adopt the computation evaluation method proposed in [30] using the MIRACL cryptographic library [36], in which the experimental platform is an Intel I7-4770 processor

Table 2: The comparison of computation cost.

Scheme	Individual-Sign	Individual-Verify	Aggregate-Verify
Gao et al. [35]	$5t_{sm}+2t_{htp}+3t_{pa}$ ≈ 17.3783 ms	$5t_{bp}+3t_{htp}$ ≈ 34.273 ms	$(n+4)t_{bp}+(2n+1)t_{htp}+3(n-1)t_{pa}$ $\approx 13.0443n+21.2429$ ms
Gong et al. (CAS 2) [27]	$3t_{sm}+2t_{htp}+2t_{pa}$ ≈ 13.9532 ms	$3t_{bp}+t_{sm}+3t_{htp}+t_{pa}$ ≈ 27.5670 ms	$(n+2)t_{bp}+nt_{sm}+2nt_{htp}+nt_{pa}$ $\approx 14.7391n+8.422$ ms
Gong et al. (CAS 1) [27]	$2t_{sm}+t_{htp}+t_{pa}$ ≈ 7.8311 ms	$3t_{bp}+t_{htp}$ ≈ 17.039 ms	$(2n+1)t_{bp}+2nt_{htp}$ $\approx 17.2340n+4.2110$ ms
Our proposed scheme	$4t_{sm}+t_{htp}+t_{pa}+t_h$ ≈ 10.2492 ms	$4t_{bp}+2t_{htp}+t_h$ ≈ 25.6460 ms	$(2n+2)t_{bp}+2nt_{htp}+(n-1)t_{pa}+nt_h$ $\approx 17.2412n+8.4149$ ms

(3.40 GHz) with 4 GB RAM, and the operating system is Windows 7. The aggregate signature scheme in BlockREV is simulated on the elliptic curve $y^2 = x^3 + x \bmod p$, where p and q are 512 bit and 160 bit prime numbers, respectively.

Some notations about cryptographic operation execution time are listed as follows.

(i) t_{bp} denotes the time for a bilinear pairing operation $e(\widehat{P}_1, \widehat{P}_2)$, where $\widehat{P}_1, \widehat{P}_2 \in \mathbb{G}_A$, $t_{bp} = 4.2110$ ms.

(ii) t_{sm} denotes the time for a scale multiplication operation $r \cdot \widehat{P}$ in the bilinear pair, where $r \in \mathbb{Z}_q^*$ and $\widehat{P} \in \mathbb{G}_A$, $t_{sm} = 1.7090$ ms.

(iii) t_{pa} denotes the time for a point addition operation $\widehat{P} + \widehat{Q}$ in the bilinear pair, where $\widehat{P}, \widehat{Q} \in \mathbb{G}_A$, $t_{sm} = 0.0071$ ms.

(iv) t_{htp} denotes the time for a hash-to-point operation in the bilinear pair, which maps a string to a point of $\mathbb{G}_A$, $t_{\mathrm{htp}} = 4.4060$ ms.

(v) t_h denotes the time for a general hash function operation, $t_h = 0.0001$ ms.

(vi) n is the number of individual signatures.

(vii) $\mathbf{l}_{\mathbb{G}}$ denotes the length of a group element in $\mathbb{G}$, $\mathbf{l}_{\mathbb{G}} =$ 128 bytes.

(viii) $\mathbf{l}_{\mathbb{Z}_q^*}$ denotes the length of $\mathbb{Z}_q^*$ element. $\mathbf{l}_{\mathbb{Z}_q^*} = 20$ bytes.

The comparison of computational costs is performed between the related aggregate signature schemes [27, 35] and our scheme in three aspects: Individual-Sign, Individual-Verify, and Aggregate-Verify. The specific comparison results are presented in Table 2.

As shown in Figure 5, for the computation cost of the individual signature algorithm, we can conclude that both Gao et al.'s scheme [35] and Gong et al.'s CAS 2 scheme [27] pay more running time than ours. For the total execution time, the percentage improvement of our algorithm over Gao et al.'s scheme and Gong et al.'s scheme (CAS 2) is about $(17.3783 - 10.2492)/17.3783 \approx 41.02\%$ and $(13.9532 - 10.2492)/13.9532 \approx 26.55\%$, respectively. Our individual signature algorithm reduces one hash-to-point operation in the bilinear pair compared with the above two schemes, which takes more than 4 ms to run. Due to the low cost of computing in the individual signing process, BlockREV is feasible for SDN environments in which switches as signers have limited computational ability.

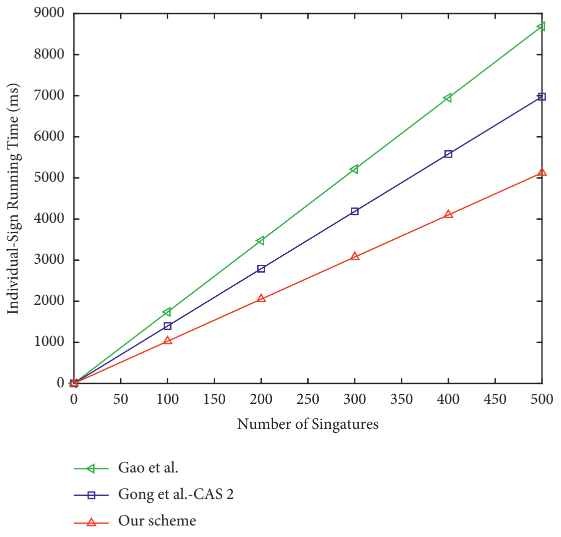


Figure 5: The comparison of individual signature time.

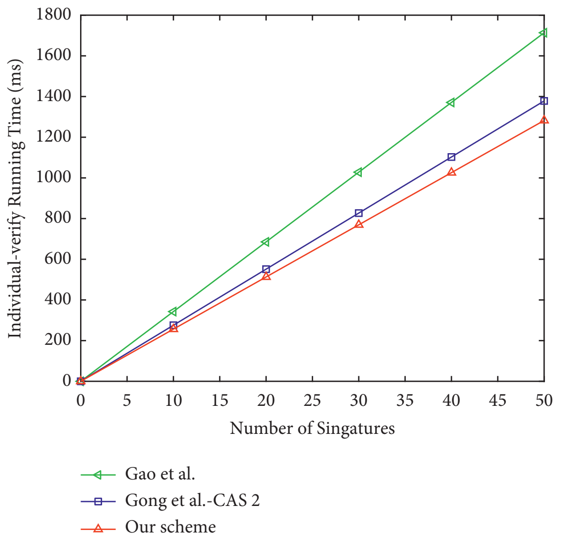


Figure 6: The comparison of individual verification time.

For the execution of the individual verification algorithm, Figure 6 shows that as the number of signatures increases, the computation cost of ours is smaller than that in Gao et al.'s scheme and Gong et al.'s scheme (CAS 2). In

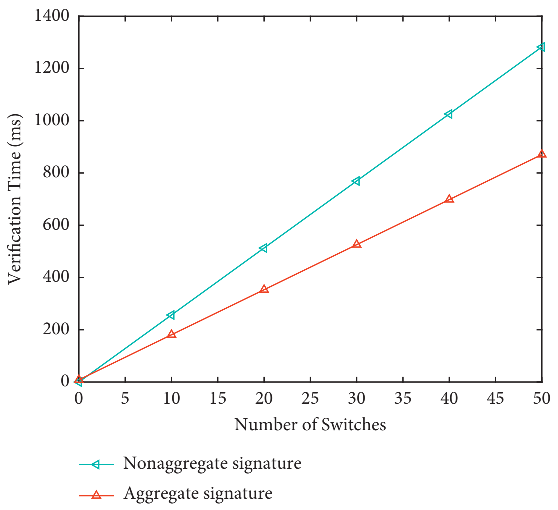


Figure 7: The comparison between non-aggregate signature and aggregate signature.

Table 3: The comparison of communication cost.

Scheme	Aggregate signature length
Shim [37]	$(3n)\mathbf{l}_{\mathbb{G}}$
Gao et al. [35]	$(2n+1)\mathbf{l}_{\mathbb{G}}$
Our proposed scheme	$(n+1)\mathbf{l}_{\mathbb{G}}$

the aspect of individual verification algorithm design, Gao et al.'s scheme has one more bilinear pairing operation and one more hash-to-point operation than our scheme, which results in an additional 2.1210 ms for each signature verification. Therefore, BlockREV is more efficient in individual signature verification and will be better suited to the controller which consumes a huge computation cost to manage its domain.

As shown in Table 2, for the execution of the aggregate verification algorithm, the computation cost of our scheme is close to that of Gong et al.'s scheme (CAS 1) [27]. The reason for the above result is that in the process of verifying aggregate signatures in our scheme, the administrator is the only legal verifier with the authority to verify aggregate signatures of different domains. Each signature contains the information about the public key of the administrator, which leads to higher computational overhead. As a manager of the whole network, the administrator has strong computing power and can bear such computing overhead.

Figure 7 shows a comparison between non-aggregate signature and aggregate signature. With the increase of the number of switches, the time of verifying the signature one by one increases rapidly which results in high computation cost. When the number of switches increases to 20, non-aggregate signature verification takes 159.6811 ms more than aggregate signature verification. The aggregate signature scheme can reduce a lot of computing overhead for the verifier; therefore, it is suitable for the administrator node in the case that the cross-domain forwarding path contains a large number of switches.

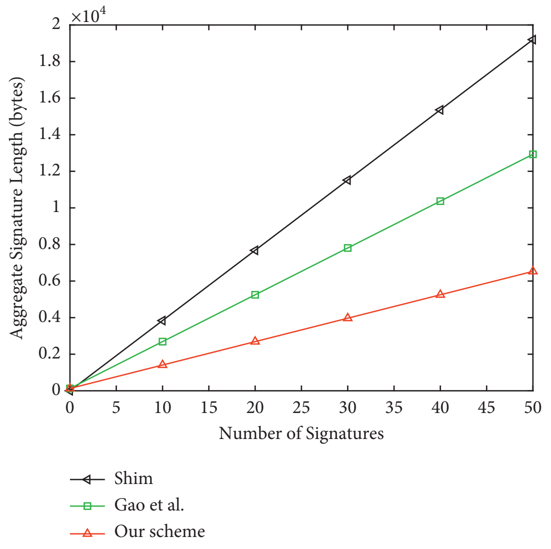


Figure 8: The comparison of aggregate signature length.

6.2. Analysis of Communication Cost. Table 3 shows the communication cost of the aggregate signature scheme and related aggregate signature schemes [35, 37]. We can see that the aggregate signature length of all schemes is increased with the number of individual signatures. The length of the aggregate signature in our scheme is $(2n-1)\mathbf{l}_{\mathbb{G}}$ less than that in Shim's scheme and $n\mathbf{l}_{\mathbb{G}}$ less than that in Gao et al.'s scheme.

As shown in Figure 8, the communication cost of our scheme is obviously smaller than that of the above two schemes; therefore, our proposed aggregate signature scheme can effectively improve the communication efficiency.

7. Conclusion

In this paper, we propose a blockchain-enabled multi-controller rule enforcement verification mechanism in SDN, called BlockREV. The mechanism adopts an address-based aggregation signature scheme with the cryptography technology and can securely store and share forwarding information with the blockchain technology. We also present the implementation results and prove the security in the random oracle model. The BlockREV mechanism can be used in many scenarios in which nodes have limited computation power, such as Internet of things and smart grid. In our future work, we will find more efficient aggregation signature constructions requiring less computation cost and design zero-knowledge proof schemes for forwarding rule verification with perfect privacy protection.

Data Availability

The experimental data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] N. Foster, N. McKeown, J. Rexford, G. Parulkar, L. Peterson, and O. Sunay, “Using deep programmability to put network owners in control,” *ACM SIGCOMM-Computer Communication Review*, vol. 50, no. 4, pp. 82–88, 2020.

[2] M. R. Belgaum, S. Musa, K. Universiti, K. Lumpur, and M. S. Mazliham, “Impact of artificial intelligence-enabled software-defined networks in infrastructure and operations: trends and challenges,” *International Journal of Advanced Computer Science and Applications*, vol. 12, no. 1, 2021.

[3] H.-z. Wang, P. Zhang, L. Xiong, X. Liu, and C.-c. Hu, “A secure and high-performance multi-controller architecture for software-defined networking,” *Frontiers of Information Technology and Electronic Engineering*, vol. 17, no. 7, pp. 634–646, 2016.

[4] B. Sengupta, Y. Li, K. Bu, and R. H. Deng, “Privacy-preserving network path validation,” *ACM Transactions on Internet Technology*, vol. 20, no. 1, pp. 1–27, 2020.

[5] P. Zhang, “Towards rule enforcement verification for software defined networks,” in *Proceedings of the IEEE INFOCOM 2017-IEEE Conference on Computer Communications*, May 2017.

[6] B. N. Astuto, M. M. Ca, N. N. Xuan, K. Obraczka, and T. Turletti, “A survey of software-defined networking: past, present, and future of programmable networks,” *IEEE Communications Surveys and Tutorials*, vol. 16, no. 3, 2014.

[7] 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, 2017.

[8] L. Qi, X. Zou, Q. Huang, Z. Jing, and P. Lee, “Dynamic packet forwarding verification in sdn,” *IEEE Transactions on Dependable and Secure Computing*, vol. 16, no. 6, p. 1, 2018.

[9] A. Khurshid, X. Zou, W. Zhou, M. Caesar, and P. Brighten Godfrey, “Veriflow: verifying network-wide invariants in real time,” in *Proceedings of the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13)*, pp. 15–27, USENIX Association, Lombard,IL,Chicago, April 2013.

[10] X. Huang, P. Shi, Y. Liu, and F. Xu, “Towards trusted and efficient sdn topology discovery: a lightweight topology verification scheme,” *Computer Networks*, vol. 170, Article ID 107119, 2020.

[11] T. H.-J. Kim, C. Basescu, L. Jia, S. B. Lee, Y.-C. Hu, and A. Perrig, “Lightweight source authentication and path validation,” in *Proceedings of the 2014 ACM Conference on SIGCOMM*, pp. 271–282, ACM, New York, NY, United States, October 2014.

[12] X. Wang, X. Chen, Y. Wang, and L. Ge, “An efficient scheme for SDN state consistency verification in cloud computing environment,” *Concurrency and Computation: Practice and Experience*, vol. 32, no. 2, 2020.

[13] A. Reyna, C. Martín, J. Chen, and E. Soler, “On blockchain and its integration with iot. challenges and opportunities,” *Future Generation Computer Systems*, vol. 88, pp. 173–190, 2018.

[14] F. Casino, T. Dasaklis, and C. Patsakis, “A systematic literature review of blockchain-based applications: current status, classification and open issues,” *Telematics and Informatics*, vol. 36, no. 11, 2018.

[15] Z. Abou 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, 2019.

[16] P. K. Sharma, S. Singh, Y. S. 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, 2017.

[17] A. Yazdinejad, R. M. Parizi, A. Dehghantanha, Q. Zhang, and K.-K. R. Choo, “An energy-efficient SDN controller architecture for iot networks with blockchain-based security,” *IEEE Transactions on Services Computing*, vol. 13, no. 4, pp. 625–638, 2020.

[18] S. Faizullah, M. Khan, A. Alzahrani, and I. Khan, “Permissioned blockchain-based security for SDN in iot cloud networks,” in *Proceedings of the 2019 International Conference on Advances in the Emerging Computing Technologies (AECT)*, February 2020.

[19] J. Weng, J. Weng, J. Liu, and Y. Zhang, “Secure software-defined networking based on blockchain,” 2019, https://www.researchgate.net/publication/333717487.

[20] J. Wang, S. Li, and S. Wei, “Identity-based cross-domain authentication by blockchain via pki environment,” in *Blockchain Technology and Application*, pp. 131–144, Springer, Singapore, 2020.

[21] A. Yazdinejad, R. Parizi, A. Dehghantanha, and K.-K. R. Choo, “Blockchain-enabled authentication handover with efficient privacy protection in SDN-based 5G networks,” *IEEE Transactions on Network Science and Engineering*, vol. 99, p. 1, 2019.

[22] N. Rajabi and J. Qaddour, “SDIoBot: a software-defined internet of blockchains of things model,” *International Journal of Internet of Things*, vol. 8, no. 1, pp. 17–26, 2019.

[23] D. Boneh, C. Gentry, B. Lynn, and H. Shacham, “Aggregate and verifiably encrypted signatures from bilinear maps,” in *Proceedings of the 22nd international conference on Theory and applications of cryptographic techniques*, pp. 416–432, Springer, Berlin,Germany, May 2003.

[24] G. Kumar, B. Singh, N. Kumar, O. Kaiwartya, and M. Obaidat, “PFCBAS: pairing free and provable certificate-based aggregate signature scheme for e-healthcare monitoring system,” *IEEE Systems Journal*, vol. 14, no. 2, pp. 1704–1715, 2019.

[25] A. Wasef, Y. Jiang, and X. Shen, “Dcs: an efficient distributed-certificate-service scheme for vehicular networks,” *IEEE Transactions on Vehicular Technology*, vol. 59, no. 2, pp. 533–549, 2010.

[26] J. Zhang, C. Hua, and G. Qin, “An efficient certificate-based signature scheme without pairings,” in *Proceedings of the Second International Workshop on Computer Science & Engineering*, IEEE, Qingdao, China, October 2009.

[27] Z. Gong, Y. Long, X. Hong, and K. Chen, “Two certificateless aggregate signatures from bilinear maps,” in *Proceedings of the 2007 8th ACIS International Conference on Software Engineering Artificial Intelligence Networking and Parallel Distributed Computing (SNPD 2007)*, August 2007.

[28] H. Shu, P. Qi, Y. Huang, F. Chen, D. Xie, and L. Sun, “An efficient certificateless aggregate signature scheme for blockchain-based medical cyber physical systems,” *Sensors*, vol. 20, no. 5, 2020.

[29] Z. Wang, “An identity-based data aggregation protocol for the smart grid,” *IEEE Transactions on Industrial Informatics*, vol. 13, no. 5, p. 1, 2017.

[30] D. He, S. Zeadally, B. Xu, and X. Huang, “An efficient identity-based conditional privacy-preserving authentication scheme

for vehicular ad hoc networks," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 12, pp. 2681–2691, 2015.

[31] J. K. Liu, J. Baek, and J. Zhou, "Certificate-based sequential aggregate signature," in *Proceedings of the Second ACM Conference on Wireless Network Security*, pp. 21–28, Association for Computing Machinery, New York, NY, USA, March 2009.

[32] Y. Wen, J. Ma, and H. Huang, "An aggregate signature scheme with specified verifier," *Chinese Journal of Electronics*, vol. 20, no. 2, pp. 333–336, 2011.

[33] S. Nakamoto, "Bitcoin: a peer-to-peer electronic cash system," 2009, https://bitcoin.org/bitcoin.pdf.

[34] N. McKeown, T. Anderson, H. Balakrishnan et al., "Openflow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[35] Y. Gao and J. Wu, "Efficient multi-party fair contract signing protocol based on blockchains," *Cryptologic Research*, vol. 5, pp. 556–567, 2018.

[36] Miracl Cryptographic Library, "Multiprecision integer and rational arithmetic C/C++ library," 2008, https://indigo.ie/mscott/.

[37] K.-A. Shim, "CPAS: an efficient conditional privacy-preserving authentication scheme for vehicular sensor networks," *IEEE Transactions on Vehicular Technology*, vol. 61, no. 4, pp. 1874–1883, 2012.