Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2016, Article ID 7145138, 8 pages
http://dx.doi.org/10.1155/2016/7145138

Research Article

A Secure Network Coding Based on Broadcast Encryption in SDN

Yue Chen,[1] Hongyong Jia,[1] Kaixiang Huang,[1] Julong Lan,[2] and Xincheng Yan[1]

[1]*State Key Laboratory of Mathematical Engineering and Advanced Computing, Zhengzhou 450001, China*
[2]*China National Digital Switching, System Engineering and Technological Research Centre, Zhengzhou 450002, China*

Correspondence should be addressed to Kaixiang Huang; kx.huang@outlook.com

Received 30 October 2015; Revised 22 May 2016; Accepted 7 July 2016

Academic Editor: Aime' Lay-Ekuakille

By allowing intermediate nodes to encode the received packets before sending them out, network coding improves the capacity and robustness of multicast applications. But it is vulnerable to the pollution attacks. Some signature schemes were proposed to thwart such attacks, but most of them need to be homomorphic that the keys cannot be generated and managed easily. In this paper, we propose a novel fast and secure switch network coding multicast (SSNC) on the software defined networks (SDN). In our scheme, the complicated secure multicast management was separated from the fast data transmission based on the SDN. Multiple multicasts will be aggregated to one multicast group according to the requirements of services and the network status. Then, the controller will route aggregated multicast group with network coding; only the trusted switch will be allowed to join the network coding by using broadcast encryption. The proposed scheme can use the traditional cryptography without homomorphy, which greatly reduces the complexity of the computation and improves the efficiency of transmission.

1. Introduction

The inflexible transport mode underlining today's network restricts the development of the networks. Its capability and structure are not well suited to the requirement of all sorts of emerging transmission services. The gap between service requirements and basic network capabilities becomes bigger. Increasing efforts have been devoted to finding more reconfigurable network, such as the software defined network (SDN) [1, 2] (e.g., OpenFlow [3, 4]) and the Flexible Architecture of Reconfigurable Infrastructure (FARI [5]).

Multicast [6] is an efficient way of disseminating information to large groups of users and plays a significant role in network services. Unfortunately, the traditional IP multicast protocols are very complex, because the router needs to not only transmit the data but also forward the control messages to get the group membership and the global network information [7]. In the new network paradigm (e.g., SDN), through separating data plane and control plane, the logically centralizing controller can control the behavior of the entire network and the router only to forwarding data. To design a scalable and secure approach for multicast is significant in the future network.

Network coding [8] is a novel transmission mechanism that allows intermediate nodes to encode the received packets, improve the capacity of multicast applications, and enhance network robustness and throughput relative to the traditional "store-and-forward" [9]. However, if there are some malicious nodes in the networks, and they forward fake vectors or invalid combinations of received vectors, the polluted vectors will be quickly spread to the other nodes and even the whole network. Only part of the multiple packets obtained by the receivers are the uncorrupt vectors. Recently, to provide a secure random linear network coding [10], most schemes need to use the homomorphic cryptographic technology to sign or hash the original data. The development of SDN brings a new method in providing the secure services.

In this paper, we propose a scheme called secure switch network coding (SSNC). Multiple multicast groups will be aggregated to one multicast group according to the requirements of services and the network status. Then, the controller will route this multicast group with network coding. Those

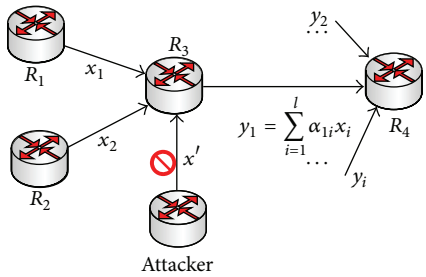

Figure 1: The pollution attacks.

switches in the same path will get the same attribute. When the data packet enters the switch in the SDN and meets the transaction's resource requirements, the switch can decrypt the received data packets and then combine the packets according to the encoding matrix. So the SSNC advances in the following aspects: (a) the controller is responsible for managing the multicast group that it can prevent attackers sending data and illegal users receiving the data; (b) the controller will authenticate and authorize the switch by only allowing trusted switches meet with the services requirements to join the path of the multicast tree; (c) the switches only forward the data according to the flow entry; and (d) we meet this challenge by using broadcast encryption and AES without homomorphism.

2. Related Work

In a typical linear network coding scenario [11], the sender first breaks the message into sequence vectors in an n-dimensional linear space $\mathbb{F}_q^n$. The sender transmits these message vectors to its neighboring nodes. The intermediate nodes will randomly and linearly combine the arriving vectors according to the local encoding matrix and then forward the new vectors to their adjacent nodes. Receivers can recover the original messages from the sufficient number of arriving packets by the encoding matrix.

When malicious nodes forward fake vectors or invalid combinations of received vectors, the receivers will have no way to decode vectors [12], and network resources are wasted. So it is crucial to prevent pollution attacks in practical applications of network coding. From Figure 1, pollution attacks cannot be mitigated by standard signatures or MACs. Because the receivers do not have the original message vectors, they cannot verify the signature of the original vectors. And there is no use to sign the entire message before transmission.

To defend against pollution attack in NC, an increasing number of researchers have proposed several novel hashing or signature schemes. Zhao et al. [13] introduced a signature scheme that breaks a file into a number of blocks viewed as vectors spanning a subspace V. Each node verifies the integrity of a received vector w by checking the membership of w in V based on the signature on V. The intermediate nodes only need to verify the signature without doing anything else. But in this scheme, the signature is too long, the public key is only used for a single file, and the sender needs to know the entire file before generating the authentication information. Therefore, Boneh et al. [14] proposed homomorphic signature schemes with better performance that both public key size and per-packer overhead are constant. This scheme signs individual vector instead of the entire subspace, but it suffers from highly expensive computational overhead due to the operations of bilinear pairing.

Yu et al. [15] proposed a probabilistic key predistribution and message authentication codes to defend against pollution attacks. The sources need to add multiple MACs to the data before forwarding them. Besides multiple nodes can verify the different parts of the message via shared secret keys. Agrawal et al. [16, 17] designed a homomorphic MAC system which allows checking the integrity of network coded data. But this system is collusion resistant up to a predetermined collusion bound c and is vulnerable to the tag pollution attacks. Based on the idea of [18], Li et al. [19] proposed a RIPPLE, a symmetric key based in-network solution for network coding authentication, which is on the basis of the homomorphic Mac and TESLA [20]. In RIPPLE, for calculating the MAC labels, the sources must know the longest path from each node to the source. Wu et al. [21] designed a key predistribution-based tag encoding scheme KEPTE. The source will generate multiple tags for each packet. Besides, the intermediate nodes will generate a new tag according to the received tags and then verify the correctness of the packet. But there must be a key distribution center in this scheme.

In the SDN architecture, the complicated secure multicast management could be separated from the fast data transmission [22, 23]. We need to verify that the switch has some authenticated attributes. So the session keys should only be bold by the switch with those attributes. The broadcast encryption can be used to distribute the session key to those switches. When a switch receives a broadcast message, it can verify that the message really comes from the controller. Because the session key of the packet and the attributes of switch are authenticated and issued by the PKG, thus the only secure switch can combine received packets according to the encoding matrix and then encrypt and forward the new packet by the flow entry. So the packet can be forwarded in the secure switches in the SDN.

3. Broadcast Encryption

The main construction of broadcast encryption for n users is as follows:

Bro.Setup(n): In this broadcast encryption scheme, a trustee is needed to generate some public information used in the following construction. The trustee chooses proper bilinear group $\mathbb{G}$ of prime order p. Choose random values: $g \leftarrow \mathbb{G}$; $\alpha \leftarrow \mathbb{Z}_p$, and set

$$g_i = g^{(\alpha^i)} \in \mathbb{G} \quad \text{for } i = 1, 2, \ldots, n, n+2, \ldots, 2n, \quad v = g^{\gamma} \in \mathbb{G}, \tag{1}$$

where a random $\gamma \in \mathbb{Z}_p$.

The public key is

$$\text{PK} = \left(g, g_1, \ldots, g_n, g_{n+2}, \ldots, g_{2n}, v\right) \in \mathbb{G}^{2n+1}. \tag{2}$$

The private key for user $i \in \{1, \ldots, n\}$ is set as $d_i = g_i^{\gamma} \in \mathbb{G}$. Note that $d_i = v^{(\alpha^i)}$.

Output the public key PK and the n private keys $d_1, \ldots, d_n$.

Bro.**Encrypt**(S, PK): Choose random values, $t \leftarrow \mathbb{Z}_p$, and set

$K = e(g_{n+1}, g)^t \in \mathbb{G}$; the value $e(g_{n+1}, g)$ can be computed as $e(g_n, g_1)$.

Then set

$$\text{Hdr} = \left(g^t, \left(v \cdot \prod_{j \in S} g_{n+1-j} \right)^t \right) \in \mathbb{G}^2, \tag{3}$$

and output the pair (Hdr, K).

Bro.**Derypt**(S, i, d_i, Hdr, PK): Let Hdr $= (C_0, C_1)$. Then, output

$$K = \frac{e(g_i, C_1)}{e\left(d_i, \prod_{j \in S, j \neq i} g_{n+1-j+i}, C_0\right)}. \tag{4}$$

A private key is only one group element in $\mathbb{G}$, and the ciphertext Hdr is only two group elements. The ciphertexts and private keys are of constant size, but the public key grows linearly in the number of users.

4. SSNC Authentication Method

We mainly use the network coding to implement the multicast application. In the SDN, a legal sender can generate and send out original packets without being aware of the encoding/decoding affairs, whereas a legal recipient can receive these packets from the network. For multicast transmission, the controller is responsible for deciding whether to use the network coding. When multiple multicast trees are aggregated to use the network coding for transmission, the controller is then responsible for routing the paths, updating the flow table of the switches, and calculating the local encoding matrix for each switch involved in the transmission. Finally, the controller generates the flow-table entries of the network coding and sends these to each switch via the interface.

Figure 2 shows that the control layer mainly consists of the control server. In the controller, the general module is responsible for basic operation and management, such as knowing the global network topology, generating and updating the flow entry, and monitoring the state awareness. To create a strategy, the general module also exchanges basic parameters with the other modules. The state-aware module monitors and analyzes the network state to promptly discover the abnormal switch nodes and current traffic distribution. The network-topology module obtains the whole network structure to provide a global view. The flow-entry module generates and updates the flow-table entries of the switches. The multicast module manages the group members, the routing of multicast trees, the decision of whether to use network coding for multicast, the selection of network-coding path, and the generation of a coding matrix based on the parameter information provided by the general module. The security module mainly generates and manages the certificates and completes the authorization and authentication of the member switches by cooperating with the other modules.

The data layer is composed of switches and users, which are only involved in data transmission. In the switch, the encoding module is responsible for encoding the data according to the local encoding matrix assigned by the controller. The decoding module only runs when the switch is connected to the receiver. It decodes according to the decoding matrix and then sends the data to the receiver. The security module stores private keys and certificates and simultaneously verifies the credibility of the data as well as its own credibility by connecting with the controller.

The network coding needs to establish the forward paths between all the senders and receivers. Besides, for traditional networks, it is difficult to know all the nodes on the path. So most researches sign the source data or the spanning vectors space and ensure the security through the source authentication in each node. If the switches on the path are all real and trusted during the whole data transmission, other switches can be effectively prevented from replaying and tampering attacks. Authentication, authorization, and integrity checks must be provided in network coding transmission. Only authenticated and authorized switches are used.

4.1. System Setup. The trusted controller runs the **Bro**.**Setup**(n) algorithm in SDN. Input the number of switches n in the SDN. Then every switch gets the system setup outputs, public PK, and its private key d_i.

4.2. Switch Session Keys Distribution. Multiple multicast groups will be aggregated to one group according to the quality of service level and the network status. Then the controller routes this group by network coding, to get the subset $S \subseteq \{1, \ldots, m\}$, and the switch $S_i = i$ belongs to this path. Input the public key PK and a subset S; the **Bro**.**Encrypt**(S, PK) algorithm outputs a pair (Hdr, K), where Hdr is the broadcast ciphertext, and the key $K \in \mathcal{K}$ is a symmetric multicast session key. Because a malicious node can fake a pair (Hdr, K), the controller chooses a collision-resistant hash function **H**, and sign the hash value of Hdr: $\sigma =$ enc(h(Hdr), Private.rsa) by the RSA algorithm. So the message (S, Hdr, σ) is broadcast to the set S:

$$\begin{aligned} &\text{Controller} \longrightarrow \\ &S_1 : \{S, \text{Hdr}, \sigma\}, \\ &S_2 : \{S, \text{Hdr}, \sigma\}, \\ &\vdots \end{aligned} \tag{5}$$

4.3. Multicast Data Forwarding. When the switch gets the broadcast message (S, Hdr, σ), it is necessary to verify the identity of the controller. So the switch decrypts the σ to get the value h(Hdr) $=$ dec(σ, Public.rsa) and computes and checks the h(Hdr) by the function **H** as the input Hdr.

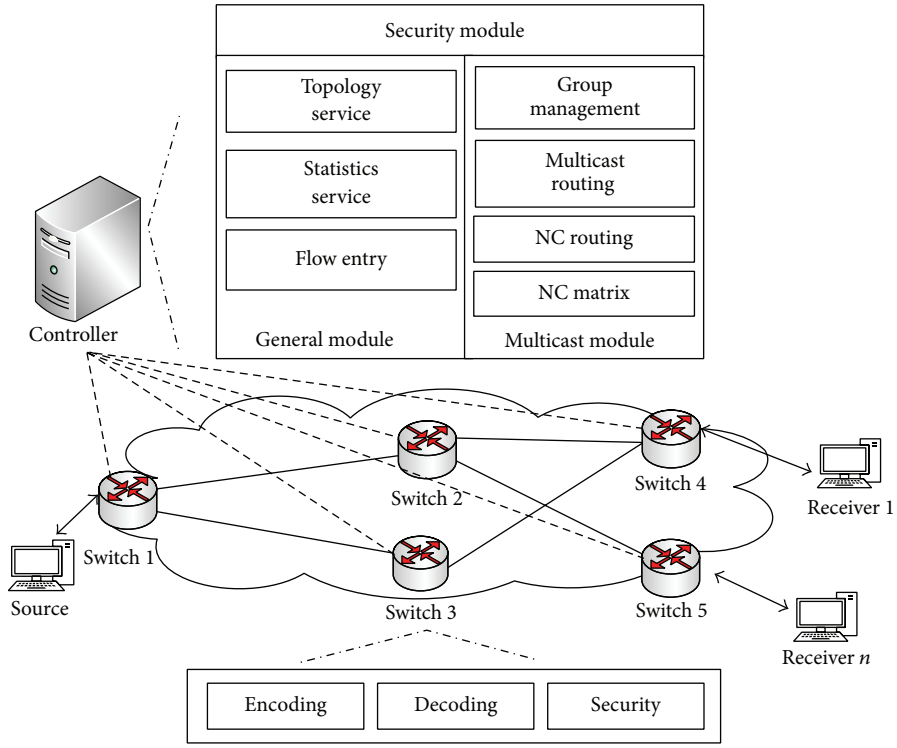

Figure 2: The scheme of SSNC.

After the message passes the verification, input a subset $S \subseteq \{1, \ldots, n\}$, the switch id $i \in \{1, \ldots, n\}$, and the private key d_i for the switch i, a header Hdr, and the public key PK, if $i \in S$. Then, the **Bro**.**Derypt**$(S, i, d_i, \text{Hdr}, \text{PK})$ algorithm outputs the multicast session key $K \in \mathcal{K}$. The key K can then be used to encrypt/decrypt the multicast data. To improve the performance of the multicast transmission, the encryption function can be embedded in hardware.

In SSNC, after the secure multicast forwarding tree is established, each switch, sender, and receiver in the multicast paths will get the session key K. For the sender, the original data is represented by a vector $V = (v_1, v_2, \ldots, v_l)$. The data vector should be encrypted by the key K, denoted as $\{v_i\}K$. The different data $\{v_i\}K$ is sent to the different next switch S:

$$\begin{aligned} &\text{Sender} \longrightarrow \\ &S_1 : \{v_1\} K, \\ &S_2 : \{v_2\} K, \\ &\vdots \end{aligned} \tag{6}$$

The switches located in the same path as the sender have the same requirement of the quality of service and security level. Only when they have the same session key K will the switches decrypt all receiving packets by using the session key K and then linearly combine the received data vectors to a new data vector y based on the predistributed encoding matrix $M_i = \{\alpha_1, \alpha_2, \ldots, \alpha_l\}^T$. After that, the switch encrypts and forwards the vector y to the next switches:

$$\begin{aligned} &y = \sum_{i=1}^{l} \alpha_i v_i \\ &S \longrightarrow \\ &S_{\text{NEXT1}} : \{y\} K, \\ &S_{\text{NEXT2}} : \{y\} K, \\ &\vdots \end{aligned} \tag{7}$$

For the receiver located in the same path as the sender, once receiving a packet, it will decrypt data using the session key K and encode the packets by network coding to get the original data.

4.4. Rekeying. When the switch is broken down or cannot meet the quality of service level or the security requirements, this switch should be removed from the subset S. The controller may need to route new path for the aggregated multicast to replace the problem. To enhance the security of the multicast, the controller also needs to rekey per certain period. Therefore, the controller is required to generate new keys for the subset S. When new keys are generated, these keys will be broadcasted to all switches. But only the switch belonging to the subset S can decrypt the new keys.

4.5. Maintenance of Multicast Path. When the multicast groups are determined, the controller is responsible for aggregating multicast groups and deciding the usage of network coding. And if it decides to use the network coding for transmission, it will route this group and provide secure and reliable switchers to construct the transmission path of multicast. It is very important to establish the trust relationship between the switch and the controller [24]. The controller can keep white lists for trusted and authenticated devices and build the multicast paths from this list. The list is dynamically changed based on anomaly/failure detection algorithms [25]. If the trustworthiness of the switch is questioned or has abnormal behavior, the switch will be reported by other switches or controllers. So this switch will be automatically quarantined by other switches and controllers and thus removed from the list.

5. Performance and Security Analysis

5.1. Security Analysis. The set of the whole network is denoted by $\mathcal{N}$, the number of all switches is n, the set of the switches in one secure multicast path is denoted by $\mathcal{S}$ ($\mathcal{S} \subset \mathcal{N}$), the number of switches is s, the number of the multicasts in the network is l, and the attacker is denoted by $\mathcal{A}$.

5.1.1. Eavesdropping Attack. An eavesdropping attacker can wiretap one or more links in the network. It also refers to the information not leaked to the unauthorized users. In the network coding, the original data will be divided into many data vectors before being sent out. And the data vectors will be combined into new data vectors. The original data can be decoded only if $\mathcal{A}$ got enough data vectors and the corresponding coding matrix. To make the eavesdropper not get the correct data, traditional network coding needs encrypting coefficients or partial data. But the encryption scheme usually needs to be homomorphic. In the SSNC, the coefficients of coding are predistributed to the trusted switch by the controller instead of being transited with the data. And the multicast packets are transmitted with the ciphertext encrypted by AES. Each of the trusted switches should be authenticated by the controller, but $\mathcal{A}$ may fake the controller to send the session K. Thereby, in the SSNC, we provide the source authentication when broadcasting the session K by the RSA.

5.1.2. Pollution Attack. This attack is usually stated by unauthorized nodes which inject polluted packets into the information flow. An attacker $\mathcal{A}$ fakes a message $\{v^*\}K^*$. Only if the switch of $\mathcal{S}$ has the fake session key K^* will it receive the data v^*. But $\mathcal{A}$ has to crack secret keys of the AES. Without cracking the AES, $\mathcal{A}$ has to fake the trusted switch to join the secure multicast group or fake the controller to distribute the session key K^* to the switches of $\mathcal{C}$. We can use the RSA technology to resolve the second problem in Section 4.2. To resolve the first problem, the switch should be authenticated and authorized by controller at the first time, when it connects to the controller. And with the system running, the controller will authenticate and authorize them at regular intervals.

The security of the whole system depends on that of the RSA, AES, and broadcast encryption used in our scheme. From the above analyses, it can be concluded that our system is capable of resisting pollution attack and the eavesdrop attack.

5.2. Performance Analysis. The communication overhead of the network coding refers to the bandwidth cost for distributing the authentication information to each switch. Without the data vectors, the overhead mainly considers two aspects, the keys and signatures. The computational overhead mainly refers to encoding/decoding the vector and verifying the signatures. The storage overhead refers to store keys, certificates, and other safety parameters.

In the traditional secure network coding, homomorphic hash, signature, and MAC are frequently used methods to defend against the pollution attack. However, they are always expensive in computation. The advantages of the network coding are only reflected in the networks consistent with faithful switches [26]. So in the SSNC, to take the advantages of the SDN, the control layer completes the complex and expensive computations. The whole secure network coding is divided into two stages. The first stage is to set up the system. The controller will select the trusted switch meeting the requirement of QoS for the set of $\mathcal{S}$ from the network $\mathcal{N}$. By using the broadcast encryption, only the secure switch can get the session K. The controller also needs route the multicast and generates and distributes the keys. In the second stage, the data layer only needs to encode and forward the data by the session K.

Each intermediate switch only needs to store its session keys and the controller public key. While the controller provides the source authentication service, the switch stores the controller public key, whose size is denoted as $|K_R|$. The session key is shared by one group for one key. We denote the size of session key as $|K_A|$. For the switch, the total storage overhead is $|K_R| + |K_A| * c$, where c is the number of the switches belonging to the secure group. For the controller, it stores a pair of keys of RSA and all the sessions keys, so that the total is $|K_R| * c + |K_A| * l$.

In the SSNC scheme, the encoding matrix and the session keys will be distributed to each switch before the stage of transferring the data. So the packets do not need to carry any more information. We denote the size of the data vector as $|v|$.

The performance of the SSNC will depend on the setup stage, as in the forwarding stage, we can encrypt the message by the hardware. So it is very important for routing the multicast by the network coding. When the group changes very frequently, the SSNC will pay more cost in the setup stage, and the performance maybe became poor. In the worst case, it should update the keys to the whole network. To reduce the change rate of the multicast paths, we aggregate the same class multicasts to one aggregated path.

To demonstrate the effectiveness of SSNC, we compared the following three schemes (Table 1).

In the scheme presented by Ho et al. in [11], the source executes signature algorithm, while the intermediate nodes execute combinations and verification algorithms. From their

Table 1: The comparison of computation, storage, and communication overhead.

Scheme	Computation	Storage	Communication
This paper	$(n_{\text{iuput}}+1)t_{\text{AES}}$ (lowest)	$\lvert K_R\rvert + \lvert K_A\rvert * c$ (controller) $\lvert K_R\rvert + \lvert K_A\rvert * l$ (switch) (lowest)	NULL (lowest)
[11]	$n_{\text{iuput}} * t_{\text{PRF}} + t_{\text{PRG}}$ (highest)	$\lvert K_{\text{HMAC}}\rvert * n * l$ (highest)	$\lvert\text{tag}\rvert + \lvert y\rvert + P$ (middle)
[7]	$n_{\text{iuput}} * t_e$ (middle)	$\lvert K_{\text{space}}\rvert * l$ (middle)	$\lvert K_{\text{space}}\rvert + \lvert y\rvert_{\text{DSA}}$ (highest)
[8]	$n_{\text{iuput}} * t_e$ (middle)	$\lvert K_{\text{NCS}}\rvert * l$ (middle)	$\lvert y\rvert_{\text{NCS}} + \lvert K_{\text{NSC}}\rvert + P$ (middle)

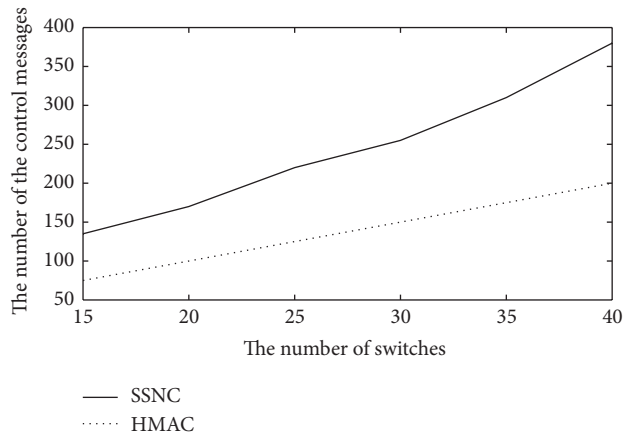

Figure 3: The number of the control messages.

experiment, it can be found that the time overhead of signature algorithm is much longer than the process of combination and verification. The intermediate nodes need to use Random Generator algorithm and Pseudo Random Function algorithm to combine the received data. The time overhead of the two algorithms is denoted as t_{PRG} and t_{PRF}. In the SSNC, the node will firstly decrypt the vectors and then combines the vectors. But in the scheme in [11], the switch firstly combines the vectors and then verifies the vectors. In this case, it is hard to tell which vector is forged. Besides, it needs to generate n keys for one multicast. Each switch distributes different keys. So it requires a complicated key management mechanism. The size of key is denoted as $|K_{\text{HMAC}}|$. Each node needs to store two sets of keys to ensure that it can act as both of the data source and receiver node. In contrast with scheme in [11], SSNC key management is more simple and effective. In the phase of data transmission, besides the original vector, it also needs to attach tags t, the extension of the vector data, and the data source id. The size of the extended vector and tag are, respectively, denoted as $|y|$ and |tag|, which is set in advance. The id of data source is a constant and denoted as P. This scheme needs to distribute keys before transmission.

In the scheme in [7], the source will find an orthogonal vector and signatures. The intermediate node will verify the signatures of which the security is based on the Diffie-Hellman problem. This computation overhead is denoted as $|t_e|$. The complexity of this algorithm is so high. This scheme uses the public key encryption scheme. The source preserves a private key and distributes the public key. Although the number of needed keys is not so much, the size of the keys is related to the size of files. We denote the size of public and secret key as $|K_{\text{space}}|$. The size of public key is $6(g+h)/gh$ times that of the file, where the file with size M is divided into g blocks and h is the size of each block. During the transmission, the main communication overhead is public key and signature vector. The vector is signed by standard algorithm, whose size is denoted as $|y|_{\text{DSA}}$.

In [8], the scheme combines linear subspace with homomorphic signature and uses constant size of public key and signature. It uses constant size of public key and signature. The internal nodes verify the signature according to public key, id, and the dimensionality of signed vector. The complexity of the calculation is similar to scheme in [7] but storage overhead is much smaller. The size of the keys is constant and denoted as $|K_{\text{NCS}}|$. Besides, the size of signature in this scheme is constant, which usually takes up $g\log_2 p$ bits or more, denoted as $|y|_{\text{NCS}}$.

Comparing with three classical solutions, the key generation is more simple and the key management is more effective in our scheme.

5.3. Network Experiment Performance. We evaluated the performance of our scheme through an experimental network. We tested SSNC on Mininet, one of the popular network simulation platforms. Each switch is connected with a receiver. Besides, one controller is connected with all switches. The controller will run the algorithm of network coding [27] according to the topology, broadcast the session keys, and update the flow entry of the switch. In the traditional secure network coding, the controller only needs to broadcast the keys. Therefore in the scheme we presented, the number of control messages increases with the number of switches in the network coding path. We emulated 5 aggregated multicast sessions. The number of the control messages is shown in Figure 3.

Before sending the multicast data packets, the switch decryption time will be dominated by the $|S|-2$ group operations, which need to compute $\prod_{j\in S, j\neq i} g_{n+1-j+i}$. This takes plenty of time. But when the switch forwards the data packets, it only needs to run the AES algorithm. The hardware will accelerate the completion of the AES algorithm. In this experiment, we used Dell desktop computers with 2.93 GHz Intel core i3 530 CPU and 4 GB main memory. The sender will send the messages to three receivers in different network topology. We timed the following operations:

(1) Broadcasting the session keys: the controller calculates the forward tree and the session K; the switch decrypts the broadcast message to get the key.

(2) Forwarding the data packets: switch decrypts and encrypts the packets using the AES.

Table 2: Time of the two stages.

Network topology	Setup	Forward
Switch = 20, receiver = 3	258.1 (ms)	4.6 (ms)
Switch = 25, receiver = 3	578.2 (ms)	4.1 (ms)
Switch = 30, receiver = 3	797.4 (ms)	4.1 (ms)

The results of multicast are shown in Table 2. We can see that the first packet of the multicast session will take more time to arrive to the receiver, but the total time of the session will take less time.

6. Conclusions

In this paper, a novel fast and secure switch network coding multicast on the software defined networks is proposed. It separates the secure management from the transmission. During the management stage, the controller routes the aggregated multicast and broadcasts the session keys. Only the secure switches meeting the requirement of quality of service can join in the multicast forwarding tree. Our scheme ensures that the switch verifies the data from other trusted switches, which greatly improves the ability to prevent the pollution and eavesdropping attacks. During data transmission, the node only focuses on how to forward the data to keep the high performance of the network coding. In order to provide more flexible and efficient multicast services, further efforts are needed to improve the system dynamics, especially on how to deal with the frequent changes of members.

Competing Interests

The authors declare that they have no competing interests.

Acknowledgments

This work is sponsored by the National Basic Research Program of China (973 Program) under Grant no. 2012CB315901 and National Natural Science Foundation of China (no. 61373006).

References

[1] B. A. A. Nunes, M. Mendonca, X.-N. Nguyen, 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, pp. 1617–1634, 2014.

[2] 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.

[3] R. Cohen, L. Lewin-Eytan, J. S. Naor, and D. Raz, "On the effect of forwarding table size on SDN network utilization," in *Proceedings of the 33rd IEEE Conference on Computer Communications (IEEE INFOCOM '14)*, pp. 1734–1742, Toronto, Canada, May 2014.

[4] A. Lara, A. Kolasani, and B. Ramamurthy, "Network innovation using open flow: a survey," *IEEE Communications Surveys and Tutorials*, vol. 16, no. 1, pp. 493–512, 2014.

[5] J. Lan, G. Xiong, Y. Hu et al., "Research on the architecture of reconfigurable fundamental information communication network," *Telecommunications Science*, vol. 31, no. 4, 2015.

[6] E. Deering S, *Multicast Routing in Internetworks and Extended LANs*, ACM, 1988.

[7] S. Keshav and S. Paul, "Centralized multicast," in *Proceedings of the 20th IEEE International Conference on Network Protocols (ICNP '12)*, p. 59, IEEE Computer Society, 2012.

[8] R. Ahlswede, N. Cai, S. Y. R. Li, and R. W. Yeung, "Network information flow," *IEEE Transactions on Information Theory*, vol. 46, no. 4, pp. 1204–1216, 2000.

[9] S.-Y. R. Li, N. Cai, and R. W. Yeung, "On theory of linear network coding," in *Proceedings of the IEEE International Symposium on Information Theory (ISIT '05)*, pp. 273–277, Adelaide, Australia, September 2005.

[10] K. Han, T. Ho, R. Koetter, M. Médard, and F. Zhao, "On network coding for security," in *Proceedings of the Military Communications Conference (MILCOM '07)*, pp. 1–6, IEEE, Orlando, Fla, USA, October 2007.

[11] T. Ho, M. Médard, J. Shi et al., "On randomized network coding," in *Proceedings of the Allerton Conference on Communication*, 2003.

[12] S. Y. R. Li, R. W. Yeung, and N. Cai, "Linear network coding," *IEEE Transactions on Information Theory*, vol. 49, no. 2, pp. 371–381, 2003.

[13] F. Zhao, T. Kalker, M. Médard, and K. J. Han, "Signatures for content distribution with network coding," in *Proceedings of the IEEE International Symposium on Information Theory (ISIT '07)*, pp. 556–560, Nice, France, June 2007.

[14] D. Boneh, D. Freeman, J. Katz et al., "Signing a linear subspace: signature schemes for network coding," in *Proceedings of the 12th International Conference on Practice and Theory in Public Key Cryptography (PKC '09)*, pp. 68–87, Springer, 2009.

[15] Z. Yu, Y. Wei, B. Ramkumar, and Y. Guan, "An efficient scheme for securing XOR network coding against pollution attacks," in *Proceedings of the IEEE INFOCOM*, pp. 406–414, IEEE, Rio de Janeiro, Brazil, April 2009.

[16] S. Agrawal and D. Boneh, "Homomorphic MACs: MAC-based integrity for network coding," in *Applied Cryptography and Network Security*, Lecture Notes in Computer Science, pp. 292–305, Springer, Berlin, Germany, 2009.

[17] S. Agrawal, D. Boneh, X. Boyen, and D. M. Freeman, "Preventing pollution attacks in multi-source network coding," in *Public Key Cryptography—PKC 2010*, vol. 6056 of *Lecture Notes in Computer Science*, pp. 161–176, Springer, Berlin, Germany, 2010.

[18] J. Dong, R. Curtmola, and C. Nita-Rotaru, "Practical defenses against pollution attacks in intra-flow network coding for wireless mesh networks," in *Proceedings of the 2nd ACM Conference on Wireless Network Security (WiSec '09)*, pp. 111–122, ACM, March 2009.

[19] Y. Li, H. Yao, M. Chen, S. Jaggi, and A. Rosen, "RIPPLE authentication for network coding," in *Proceedings of the IEEE INFOCOM*, pp. 1–9, San Diego, Calif, USA, March 2010.

[20] H. Krawczyk and R. Canetti, "The TESLA broadcast authentication protocol," *Rsa Cryptobytes*, vol. 20, no. 2, article 2002, 2002.

[21] X. Wu, Y. Xu, C. Yuen, and L. Xiang, "A tag encoding scheme against pollution attack to linear network coding," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 1, pp. 33–42, 2014.

[22] U. C. Kozat, G. Liang, and K. Kokten, "On diagnosis of forwarding plane via static forwarding rules in Software Defined Networks," in *Proceedings of the 33rd IEEE Conference on Computer Communications (IEEE INFOCOM '14)*, pp. 1716–1724, IEEE, Toronto, Canada, May 2014.

[23] R. Cohen, L. Lewin-Eytan, J. Naor S, and D. Raz, "On the effect of forwarding table size on SDN network utilization," in *Proceedings of the IEEE Conference on Computer Communications (INFOCOM '14)*, pp. 1734–1742, IEEE, Toronto, Canada, April-May 2014.

[24] D. Kreutz, F. M. V. Ramos, and P. Verissimo, "Towards secure and dependable software-defined networks," in *Proceedings of the 2nd ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking (HotSDN '13)*, pp. 55–60, ACM, Hong Kong, August 2013.

[25] Z. Yan and C. Prehofer, "Autonomic trust management for a component-based software system," *IEEE Transactions on Dependable & Secure Computing*, vol. 8, no. 6, pp. 810–823, 2011.

[26] S. Yao, J. Chen, R. Du, L. Deng, and C. Wang, "A survey of security network coding toward various attacks," in *Proceedings of the 13th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom '14)*, pp. 252–259, IEEE Computer Society, Beijing, China, September 2014.

[27] S. Jaggi, P. Sanders, P. A. Chou et al., "Polynomial time algorithms for multicast network code construction," *IEEE Transactions on Information Theory*, vol. 51, no. 6, pp. 1973–1982, 2005.