Computer Science and Information Systems 13(2):531–545 DOI: 10.2298/CSIS160222013H

Fast Multicast Scheme with Secure Network Coding in Cloud Data Centers

Kaixiang Huang[1], Yue Chen[1,2], Hongyong Jia[1], Julong Lan[2], Xincheng Yan[1], and Zhiwei Wang[3]

[1] Sciences State Key Laboratory of Mathematical Engineering and Advanced Computing, 450001 Zhengzhou, China
kx.huang@outlook.com

[2] China National Digital Switching System Engineering and Technological Research Centre 450001 Zhengzhou, China
Cyue2008@126.com

[3] Department of Computer Sciences of the University of Hongkong Hongkong, China

Abstract. Multicast is widely applied in cloud data centers. Because intermediate nodes can encode the packets, network coding improves the capacity and robustness of multicast applications. However, this system is vulnerable to pollution attacks. Existing schemes mainly focus on homomorphic cryptographic technologies against such attacks. However, the homomorphic cryptographic technology introduces complicated key management and calculation and storage overhead. This paper proposes a novel, fast, and secure network-coding multicast on software-defined networks. This scheme separates the complicated secure multicast management from fast data transmission. In the control layer, when users and switches try to join the secure multicast, they are authenticated and authorized by the controller. Only trusted nodes can join the forwarding paths. In the data layer, the trusted nodes only forward the data. The proposed scheme can use traditional cryptography without homomorphy; thus, it greatly reduces computation complexity, improves transmission efficiency, and thwarts pollution and eavesdropping attacks.

Keywords: cloud data center, multicast, secure network coding, software-defined networks.

1. Introduction

Many applications have been implemented in the cloud data center, an important cloud-computing infrastructure [1]. Cloud services such as pushing information, distributing files [2], and deploying replicated data [3], among others, often need to transfer data from one point to one or more points. Therefore, multicast technology is widely adopted to conserve the bandwidth of the data center networks [4][5]. Network coding [6] 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” mechanism [7].

In a typical linear network-coding scenario [8], the sender has to break the original data into a set of vectors .Then, the intermediate nodes randomly and linearly combine the arriving vectors according to the local encoding matrix and then forward the new vectors to their downstream nodes. When the recipients receive a sufficient number of packets, they can obtain the original data by decoding these vectors. However, if some malicious nodes are present in the network and they forward fake vectors or invalid combinations of received vectors, the polluted vectors will be quickly spread to the other nodes even in the whole network. Only some of the multiple packets obtained by the recipients are uncorrupt vectors. In this case, the recipients have no way of decoding the vectors [9], and network resources are wasted. Therefore, preventing pollution attack is very important to make network coding more practical. Fig. 1 shows that preventing pollution attack is difficult using the standard signatures or message authentication codes (MACs). Because intermediate nodes may encode the data vectors, the signatures or MACs may be damaged. The signatures or MACs of the original vectors cannot be verified by the recipients.

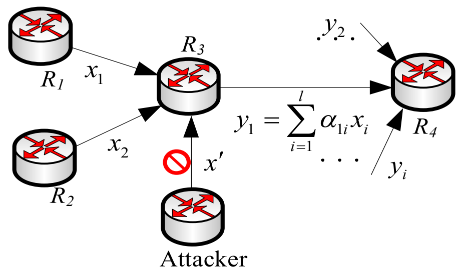

Fig. 1. Pollution attacks

In this paper, we propose a scheme called secure switch network coding (SSNC) to prevent against pollution attack. SSNC can use the traditional cryptographic technology without homomorphy to prevent pollution attacks, which reduces the complexity of the calculation and transmission delay.

In section 2 we discuss prior work defending the pollution attacks in the network coding, and describe the advantages of our scheme. In section 3, we introduce our scheme how to design and implement in the SDN. In the section 4, we describe the details of the SSNC protocol. In the section 5, we analysis the performance and security of the SSNC scheme. Finally, a conclusion is made in Section 6.

2. Related Work

To defend against pollution attack in network coding, an increasing number of researchers have proposed several novel hashing or signature schemes. Zhao et al. [10] introduced a signature scheme based on the linear subspace. At the sender side, the file is broken into a set of vectors $v_1, ..., v_m$, and these vectors can span subspace V. In the intermediate nodes, these vectors are linearly combined to obtain new vector w. If vector w belongs to subspace V, it can be accepted. The intermediate nodes only verify the signature without doing anything else. However, in this scheme, the sender must find the orthogonal vectors for each vector $v_1, ..., v_m$ then obtain the signature using the standard signature scheme.

Thus, the signature is very long, and the public key is only used for a single file. The process will become complex when multiple multicast groups are present. Therefore, Boneh et al. [11] proposed a homomorphic signature scheme in which they can sign the linear subspace using a constant public-key size. This scheme signs an individual vector instead of the entire subspace; however, it suffers from highly expensive computational overhead due to bilinear pairing operations.

Yu et al. [12] proposed a probabilistic key pre-distribution scheme to prevent pollution attacks. The sources need to add multiple MACs to the data before forwarding them, and multiple nodes can verify the different parts of the message via shared secret keys. Agrawal et al. [13][14] designed a homomorphic MAC scheme, which allows each node to check the integrity of a data vector. However, this system is vulnerable to collusion and tag pollution attacks. To defend against the tag pollution attack, Li et al. [15] proposed the RIPPLE based on the idea of Ref. [16]. This scheme provides a symmetric key solution using the homomorphic MAC algorithms and TESLA [17]. However, to calculate the MAC labels in RIPPLE, the sources must take the longest path from each node to the source as the input. Wu et al. [18] proposed KEPTE, which is a key pre-distribution-based tag encoding scheme. The source generates multiple tags for each packet. The intermediate nodes generate a new tag according to the received tags and then verify the correctness of the packet. However, this scheme requires a key distribution center.

Recently, most schemes have used the homomorphic cryptographic technology to sign or hash the original data to provide a secure random linear network coding. The development of the software-defined network (SDN) [19][20] and the reconfigurable networks [21] presents a new idea of providing secure services. In the SDN architecture, the complicated secure multicast management can be separated from the fast data transmission [22][23]. In the current paper, we propose a scheme called SSNC to prevent against pollution attack. SSNC possesses advantages in the following aspects: (a) the controller is responsible for managing the multicast group; thus, it can prevent attackers from sending data and illegal users from receiving data; (b) the controller authenticates and authorizes the switch by only allowing a trusted switch to join the path of the multicast tree; (c) the switches only encode/decode and forward the data according to the flow entry; and (d) SSNC can use the traditional cryptographic technology without homomorphy to prevent pollution attacks, which reduces the complexity of the calculation and transmission delay.

3. Design and Implementation of SSNC

We mainly use network coding to implement the multicast application [24][25]. 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 in 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 in 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.

Fig 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.

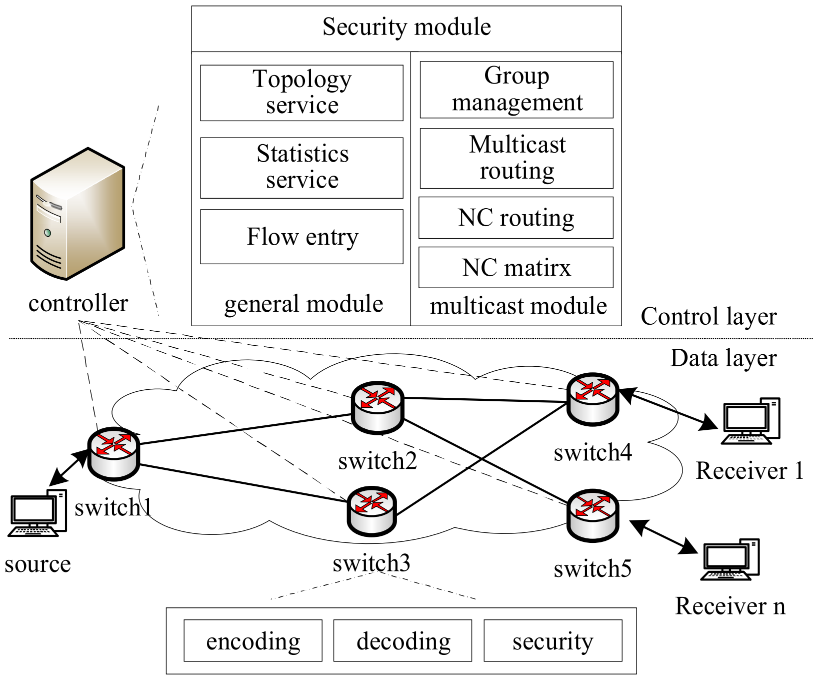

Fig. 2. SSNC scheme

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 complete multicast communication process is divided into the following steps:

Step 1: In the setup stage, the controller authenticates the true identity of all users and switches and builds the trusted switch and user lists.

Step 2: The source and the receivers send request packets to join with or quit from the group. The controller verifies each source and receiver and then computes the forwarding tree and the security policies according to the network status, the known trusted switch list, and the group members. Finally, the controller sends the strategy to the switches, which are in the path of the forwarding tree.

Step 3: The switches update the flow entries of the group.

Step 4: While the source sends the data, the switch forwards the data according to the flow entry. The receivers obtain the data when the session is finished.

Step 5: During the transmission, the controller dynamically adjusts the tree and the policies.

4. Secure Switch Multicast Scheme

Network coding needs to establish the forwarding paths for the multicast group. In addition, knowing all the nodes in the path is difficult for traditional networks [26]. Therefore, most research works sign the source data or the spanning vector space and ensure security through source authentication in each node. If the switches in the path are all real and trusted during the whole data transmission, they can effectively prevent other switches from replaying and tampering attacks. Authentication, authorization, and integrity checks must be provided in the network-coding transmission. Only authenticated and authorized switches must be used.

4.1. Initialization

The control node in the whole SDN is a trusted controller (denoted as *CN*). It maintains the list of group members by deciding which members can be added to the secure multicast group and what their permissions (such as create, join, and delete) are. *CN* has a pair of keys for the public key cryptosystem, which is used to issue a certificate for group members. When an entity wants to join a secure multicast, it needs to provide its public key to *CN* to verify its true identity. The authenticated and approved entity will receive a certificate consisting of the IP address, multicast group address, public key, permission, time stamp, and lifetime. As an example, the following is a certificate issued to user U.

$$CERT_U = [IP_U, MC, K_{+U}, Perm, TS, Life]\, K_{-CN}$$

where $CERT_U$ is the certificate, IP_U is the IP address of U, MC is the multicast address, K_{+U} is the public key of U, $Perm$ is the permission of U, TS is the time stamp, and $Life$ is the lifetime of this certification. K_{-CN} is the private key corresponding to the public key of the controller, and $[message]\,K_{-CN}$ denotes the signature of the message using K_{-CN}.

Simultaneously, if the multicast members are determined, *CN* is responsible for routing the multicast and deciding whether to use the network coding. When it decides to use the network coding for transmission, *CN* provides secure and reliable switches to construct the transmission path of the multicast. It can prevent many attacks and only allows the trusted nodes to join the forwarding path [27][28]. Establishing a trust relationship between the switch and CN is very important [29]. *CN* maintains a white list of the trusted and authenticated devices and builds the multicast paths from this list. The list is dynamically changed based on anomaly or failure detection algorithms [30]. If the trustworthiness of the devices are questioned or shows abnormal behavior, it will be reported by other switches or controller. Once the trustworthiness of a switch drops below the threshold, the switch will automatically be quarantined by other switches and controller. Then, this switch is removed from the list. Trusted *CN* distributes a certificate for each secure and trusted switch as follows:

$$CERT_R = [IP_R, MC, K_{+R}, TS, Life]\, K_{-CN}$$

4.2. Group Creation

When source *S* wants to create a secure multicast network, first, it needs to send an apply message to *CN* for a certificate. The message consists of the IP address, public key, multicast address, and permissions desired.

$$S \rightarrow CN : [IP_S, K_{+S}, MC, Perm]\, K_{-S}$$

When *CN* receives the application from the source, it will check the truth of source *S* according to the public key and IP of *S*. Then, it checks whether *S* has the permission to create a secure multicast. If the above checks are successful, it issues a certificate to *S* with the private key of *CN* and sends it to *S*.

$$CN \rightarrow S : CERT_S = [IP_S, MC, K_{+S}, TS, Life] K_{-CN}$$

When the *CN* multicast management module has issued a certificate for the source distribution and has created a group, it can accept the application of the group members to join the multicast tree. In SSNC, some control messages are in place to join or quit the multicast, which are called *Join Request* (*JR*), *Join Acknowledgement* (*JA*), *Controller Request* (*CR*), *Controller Acknowledge* (*CA*), and *Quit Notice* (*QN*). To ensure that no authorized users and switches can be added to the multicast tree, *JR* and *JA* messages need to be signed. However, the *QN* message cannot allow new branches to be added to the multicast tree; thus, no signature is needed. The *CR* and *CA* messages are used to prevent attacker replay of the *JR* and *JA* messages, and they must be signed.

The establishment process of the multicast tree is introduced as follows. When a receiver wants to join the multicast tree, it sends a *CR* message signed by the sender containing random number N_U and the sender certificate. Then, the message is directly transmitted to *CN* via the nearest switch.

$$U \rightarrow CN : [CR, CERT_U, N_U]\, K_{-U}$$

When *CN* receives the *CR* message, it verifies the message and the certificate. Once the verification is successful, it directly sends the identification information *CA* to the receiver signed by *CN*, which contains the certificate $CERT_{CN}$ of *CN*, random number N_U of the receiver, and random number N_{CN} generated by *CN*.

$$CN \rightarrow U : [CA, CERT_{CN}, N_U, N_{CN}]\, K_{-CN}$$

When the receiver receives the identification message from CN, it sends the joining message JR to CN, which contains certificate $CERT_U$ of the receiver and random number N_{CN} provided by CN.

$$U \rightarrow CN : [JR, CERT_U, N_{CN}]\, K_{-U}$$

Once CN receives joining message JR, CN verifies it. Further, CN will calculate the multicast forwarding tree based on the trusted switch and update all the flow entry and security policy of the trusted switch. The set of selected trusted switches is denoted as R. The update information includes the public key of CN, multicast ID, initial sequence SEQ, and sharing key K_B of the forward tree. Finally, CN needs to sign the message and sends an identification message to user U. Meanwhile, the sharing key of the tree can choose to use the classical symmetric encryption scheme such as the Advanced Encryption Standard (AES), which has a faster speed compared with the public-key cryptography.

$$\begin{aligned} CN \rightarrow R : & [JA_{R1}, CERT_{CN}, \{K_B, ID, SEQ,\}\, K_{+R1}]\, K_{-CN}, \\ & [JA_{R2}, CERT_{CN}, \{K_B, ID, SEQ,\}\, K_{+R2}]\, K_{-CN}, \cdots \\ CN \rightarrow U : & [JA, CERT_{CN}, N_U\, \{K_B, ID, SEQ\}\, K_{+U}]\, K_{-CN}. \end{aligned}$$

4.3. Multicast Data Forwarding

In the SSNC protocol, the multicast data mainly consist of two parts. One part is the data actually needed to be sent, denoted as v_i. The other part is the control information, which is encrypted by the sharing multicast session key K_B. After the secure multicast forwarding tree is established, each switch, sender, and receiver in the multicast paths can obtain session key K_B, initial sequence SEQ, and the checksum of the data.

For the sender, the original data are represented by vector $V = (v_1, v_2, ..., v_l)$, and the different data vector v_i is sent to the different next switch R_{NEXT}.

$$\begin{aligned} R \rightarrow R_{NEXT1} : & \{v_1\}, \{ID, SEQ, CS(\{v_1\})\}\, K_B, \\ R_{NEXT2} : & \{v_2\}, \{ID, SEQ, CS(\{v_2\})\}\, K_B, \cdots \end{aligned}$$

For switch R, which is located in the same path as the source, once packets are being received, it decrypts the control part of the packet using session key K_B, checks the ID and the SEQ of the multicast, and checks the checksum of the data part of the packet. If all checks are successful, all the received data vectors are linearly combined to new data vector y based on encoding matrix M. Then, the new data vector and the control information are sent to the next switcher, where ID indicates which multicast does the packet belong to, SEQ indicates which generation of the network coding the packet belongs to, and the checksum is verified against the modified data.

$$y = \sum_{i=1}^{l} \alpha_i x_i.$$

$$\begin{aligned} R \to R_{NEXT1} &: \{y\}, \{ID, SEQ, CS(\{y\})\} K_B, \\ R_{NEXT2} &: \{y\}, \{ID, SEQ, CS(\{y\})\} K_B, \cdots \end{aligned}$$

For receiver U, which is located in the same path as the source, once it receives a packet, it decrypts the control part of the data using session key K_B and checks the ID, SEQ, and checksum of the data. If all checks are successful, U will encode the packet using network coding to obtain the original data.

4.4. Key Update

When members change or some nodes are no longer trusted, session key K_B should be updated in time; otherwise, malicious nodes will forge the data using the old key and then launch a pollution attack. Meanwhile, the life cycle of the key also requires regular updating. Therefore, *CN* must generate new keys and reset SEQ. After a new key is generated, the old key and new SEQ are encrypted by the public keys of each member and switch, which are then encrypted by the private key of *CN*. Finally, they are sent to all the switches and members of the paths.

$$\begin{aligned} CN \to R &: [IP_{R1}, \{K_{BNew}, K_{BOld}, SEQ\} K_{+R1}] K_{-CN}, \\ &\quad [IP_{R2}, \{K_{BNew}, K_{BOld}, SEQ\} K_{+R2}] K_{-CN}, \cdots \\ CN \to U &: [IP_{U1}, \{K_{BNew}, K_{BOld}, SEQ\} K_{+U1}] K_{-CN}, \\ &\quad [IP_{U2}, \{K_{BNew}, K_{BOld}, SEQ\} K_{+U2}] K_{-CN}, \cdots \end{aligned}$$

4.5. Maintenance of Multicast Path

The change in the multicast path is mainly caused by some nodes that quit or join the group. *CN* can automatically delete the nodes that cannot be reached due to fault or to the untrusted nodes according to the detection algorithm (such as expiration of the certificate or abnormal behavior). These nodes must be deleted from the security path list. To replace the problem node, *CN* calculates a new path with the known trusted nodes, updates the key, and encodes the matrix. When a trusted user wants to quit the multicast group, it sends a *QN* message to *CN*. Once *CN* receives the *QN* message, it adjusts the multicast path and simultaneously updates the key and the encoding matrix. To reduce the cost of updating the encoding matrix and the key, the nodes with high trustworthiness, determined by historical records, should be mainly selected in building the path. When a new user (or switch) is added to the secure multicast group, its credibility must first be ensured by authentication, and only an authenticated one can be added into the secure multicast path list to compute the path and update the key and encoding matrix.

5. Performance and Security Analysis

5.1. Security Analysis

The set of the whole network is denoted as $\mathscr{N}$. The set of nodes in the secure multicast path is denoted as $\mathscr{C}$ ($\mathscr{C} \subset \mathscr{N}$). The number of multicasts in the network is denoted as l, and the attacker is denoted as $\mathscr{A}$.

Confidentiality means that information must not be leaked to unauthorized users. In network coding, the original data are divided into many data vectors before being sent. The data vectors will be combined into new data vectors. The original data can be decoded only if $\mathscr{A}$ obtains sufficient data vectors and the corresponding coding matrix. To prevent an eavesdropper from obtaining the correct data, the traditional network coding encrypts the coefficient or partial data. However, the encryption scheme usually is homomorphic. In the SSNC, the coding coefficients are not transmitted with the data but distributed to the trusted node by the RSA. To obtain the coefficient, must obtain the private key of the trusted node or successfully fake the trusted node certification; thus, this mechanism can provide confidentiality.

Each of the trusted nodes is authenticated by the public key infrastructure. However, $\mathscr{A}$ may fake the source or the trusted node to send and forward data. To solve this problem, we provide the source authentication in SSNC by combining the hash with symmetric encryption. When the multicast service requires source authentication, the data part is added by the *IP* of the source. The control part of the packet can be faked only if $\mathscr{A}$ obtains the session key. However, $\mathscr{A}$ can tamper the data part with the correct control part; thus, we use the hash function to protect against this attack. Non-repudiation is provided by adding the source *IP* encrypted by its private key to the data part, for example,

$$\begin{aligned} R \to R_{NEXT1} &: \{y, [IP_S]\, K_{-S}\}, \{ID, SEQ, CS(\{y, [IP_S]\, K_{-S}\})\}\, K_B, \\ R_{NEXT2} &: \{y, [IP_S]\, K_{-S}\}, \{ID, SEQ, CS(\{y, [IP_S]\, K_{-S}\})\}\, K_B, \cdots \end{aligned}$$

Integrity refers to that information in the process of transmission cannot be changed without authorization. For example, the pollution attack tampers the data by faking or adding some invalid data. An attacker $\mathscr{A}$ fakes a message $\{v^*\}$, $\{ID, SEQ, CS(\{v^*\})\}K^*$. Once the node of $\mathscr{C}$ receives this data and decodes the control part, the message v^* will be accepted. However, $\mathscr{A}$ has to crack secret keys of the AES. Without cracking the AES, $\mathscr{A}$ has to fake the trusted nodes to join the secure multicast group. It sends the message $U^* \to CN : [JR, CERT_{U^*}, N_{CN^*}]\, K_{-U^*}$ to CN. The private key should be known, as well as the certificate of which the security depends on the RSA. In another case, when the attackers eavesdrop on the link between CN and the trusted node, they will get the message: $CN \to R : [JA_R, CERT_{CN}, \{K_B, ID, SEQ,\ \}K_{+R}]K_{-CN}$. Obtaining the CN public key is easy, but obtaining the node private key is difficult.

To obtain the data of the multicast, attacker $\mathscr{A}$ eavesdrops on the joint messages sent by the trusted node, which then returns the false key to the node. For example, a source wants to send a joint message to CN, The attacker eavesdrops on the message and returns the message: $CN^* \to S : CERT_S^* = [IP_S, K_{+S}, MC, Perm, TS, Life]K_{-CN^*}$ to the source. However, forging *CERT* without the *CN* private key, which depends on the RSA algorithm, is difficult.

Sometimes, although the messages: $[JA_R, CERT_{CN}, \{K_B, ID, SEQ,\}K_{+R}]K_{-CN}$ and $[JA, CERT_{CN}, N_U\,\{K_B, ID, SEQ\}K_{+U}]K_{-CN}$ are eavesdropped, obtaining K_B or forging the messages remains difficult.

The security of the whole system depends on the encryption algorithms used in our scheme. From the above analyses, we can conclude that our system is capable of resisting pollution and eavesdrop attacks.

5.2. Performance analysis

The communication overhead of network coding refers to the bandwidth cost to distribute the authentication information to each node. The overhead mainly considers two aspects: keys and signatures. The computational overhead mainly refers to encoding/decoding the vector and verifying the signatures. The storage overhead refers to the storing of the keys, certificates, and other safety parameters.

SSNC is composed of two stages. The first stage is the setting up of the system. CN selects the trusted nodes for the set of $\mathscr{C}$ from the set of $\mathscr{N}$. Using RSA, CN authorizes and authenticates the nodes. We denote t_R as a running time for finishing a RSA algorithm. In the setup stage, interactions between CN and the nodes occur two times; thus, the total time is $2t_R$. During the building of a secure multicast group, interactions between CN and the nodes also occur two times, but running the RSA algorithm requires five times. This type of overhead does not affect the time of transmission. For the second SNNC stage, each intermediate node must decrypt and verify the control part of the packet. If the decrypting and verification algorithms are very complex, the efficiency of the data transmission will significantly be reduced. Therefore, we use the AES algorithm to encrypt and decrypt the control part of the packet by denoting the time of completing the AES algorithm as t_{AES}. Before combing the data vectors, we verify the vectors without tampering them using the hash algorithm and denote the time of completing the hash algorithm as t_{hash}. Before the new vector is forwarded to the next node, the node also needs to generate a new hash value using hash and encrypts the new control part of the packet using AES. If a node receives n_{iuput} packets, the total computation overhead of this node is $(n_{iuput}+1)(t_{AES}+t_{hash})$.

For each intermediate node,two types of keys are needed for storage. One is the public key, whose size is denoted as $|K_R|$. The intermediate node should store its private key and the CN public key. Whereas the system needs to provide the source authentication service, the node needs to store the source public key. The other is the session key, where one key is shared by one group. We denote the size of the session key as $|K_A|$. FFor the intermediate node, the total storage overhead is $3|K_R|+|K_A|\cdot l$. CN stores the public nodes of the whole $\mathscr{C}$ and all the session keys; therefore, the total overhead is $|K_R|\cdot c+|K_A|\cdot l$.

In the SSNC scheme, the encoding matrix and the session keys are distributed to each node before transferring the data. Thus, the packets do not have to carry too much information. We denote the size of the data vector as $|v|$, and the size of the checksum as $|v|_{hash}$ (which is a constant, e.g., 16 bytes). The sizes of multicar ID and SEQ, denoted as p, are also constant as well as the total of the control part $||v|_{hash}+P|_{AES}$.

To demonstrate the effectiveness of SSNC, we compared the following three schemes.

In the scheme presented by Agrawal et al. in Ref. [13], the source executes signature algorithm, whereas the intermediate nodes execute the combination and verification algorithms. From their experiment, they found that the time overhead of the signature algorithm is much longer than that of the combination and verification process. The intermediate nodes need to use the random generator and pseudo-random function algorithms to combine the received data. The time overhead of the two algorithms are denoted as t_{PRG} and t_{PRF}, respectively. In SSNC, the node first verifies the vectors and then combines them. However, in the scheme in Ref. [13], the node first combines the vectors and then verifies them. Therefore, determining which vector is forged is very hard. Further, the scheme needs to generate n keys for one multicast. For each node, it distributes a different key. Thus, it requires a complex key management mechanism. The size of the key is denoted as $|K_{HMAC}|$. Each node needs to store two sets of keys to ensure that it can act as both data source and receiver node. In contrast to the scheme in Ref. [13], the SSNC key management is more simple and effective. In the data transmission phase, in addition to the original vector, it also needs to attach tags t, extension of the vector data, and data source ID. The sizes of the tags and the extended vector are respectively denoted as $|tag|$, which is set in advance, and $|y|$. The ID of the data source is a constant and denoted as P. Further, this scheme needs to distribute the keys before transmission.

Table 1. Comparison of the computation, storage, and communication overhead

Scheme	Computation	Storage	Communication
This work	$(n_{iuput}+1)(t_{AES}+t_{hash})$	$CN: \|K_R\|\cdot c+\|K_A\|\cdot l,\ R: \|K_A\|\cdot l$	$\|\|v\|_{hash}+P\|_{AES}$
Agrawal	$n_{iuput}\cdot t_{PRF}+t_{PRG}$	$\|K_{HMAC}\|\cdot n\cdot l$	$\|tag\|+\|y\|+P$
Zhao	$n_{iuput}\cdot t_e$	$\|K_{space}\|\cdot l$	$\|K_{space}\|+\|y\|_{DSA}$
Boneh	$n_{iuput}\cdot t_e$	$\|K_{NCS}\|\cdot l$	$\|K_{NCS}\|+\|y\|_{NCS}+P$

In the scheme in Ref. [10], the source finds an orthogonal vector and the signatures. The intermediate node verifies the signatures whose security is based on the DiffieCHellman problem. This computation overhead is denoted as $|t_e|$. The complexity of this algorithm is much higher. This scheme uses the public key encryption scheme. The source preserves the private key and distributes the public key. Although the number of needed keys is not very large, the size of the key is related to the size of files. We denote the size of the public and secret keys as $|K_{space}|$, and 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 the public key and signature vector. This scheme signs the vector using standard algorithm, and the size is denoted as $|y|_{DSA}$.

The scheme in Ref. [11] combines the linear subspace with a homomorphic signature. It uses a constant size of the public key and signature. The internal nodes verify the signature according to the public key id and the dimensionality of the signed vector. The complexity of the calculation is similar to that of the scheme in Ref. [10], but the storage overhead is much smaller. The size of the keys is constant and is denoted as $|K_{NCS}|$. In addition, the size of the signature in this scheme is constant, which usually takes up $g\log_2 p$ bits or more, denoted as $|y|_{NCS}$.

By comparing the three classical solutions, we can find that in our scheme, the key generation is simpler, the key management is more effective, and additional information for the packet is reduced. Although the intermediate nodes need to encrypt/decrypt the packets, the AES is very quick, and the length of control information is constant.

5.3. Simulation Experiment

We verify the performance of SNNC by simulation experiments. We mainly use Mininet [31] to carry out the SDN, where each switch is connected to a receiver. Each switch is also connected to the controller, and the system contains one sender and five receivers. We generate different network topologies using Mininet and generate the encoding matrix and encoding nodes using the network-coding algorithm [32]. The controller needs to run the RSA authentication algorithm, generate the AES keys, and distribute the keys to each node in the forwarding path. Each node encodes the packets with a simple XOR operation.

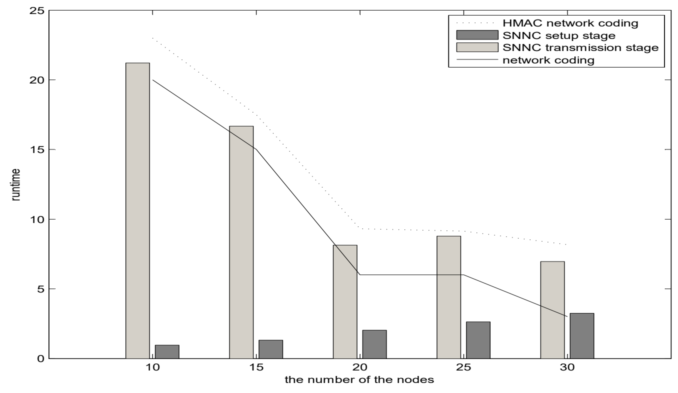

Fig. 3. Runtime of the SSNC

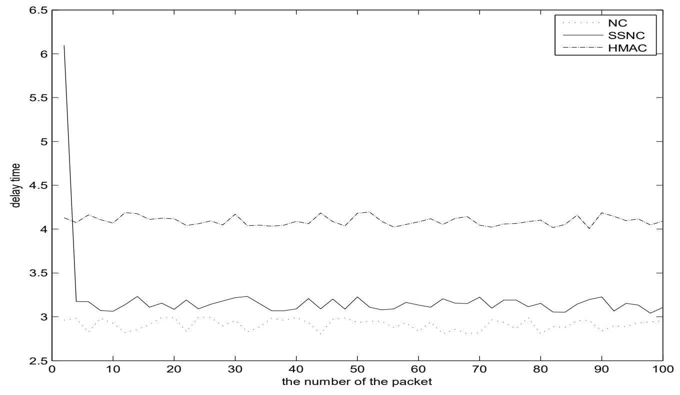

Fig. 4. Delay time of the SSNC

In the experiment, we use a Dell desktop computer, which has a 2.93 GHz Intel core i3 530 CPU with 4 GB main memory and runs on Ubuntu Linux. First, the sender sends the packets at the calculated maximum flow, and each packet is 1 k bytes in size. We record two time types: (a) the setup stage (the time for the controller to authenticate the switches and the time for the switches to update the entries) and (b) the transmission stage (the data-forwarding time). Fig. 3 shows that when the nodes in the network increase, the maximum flow of the network and the setup time also increase. However, the total time of the multicast is shorter, and the total multicast time is less than that of the Homomorphic MAC (HMAC) scheme proposed by Agrawal et al. [13]. Second, the data source continuously sends the data. We record the delay time of each packet. Fig. 4 shows our recorded delay curves of the 100 packets. SNNC requires more time to set up. Therefore, the first packet requires more time to arrive at the receivers. However, the total delay time is less than that of the HMAC scheme.

6. Conclusions

In this paper, we have proposed a novel and fast SSNC multicast on the SDNs. We separate the secure management from the transmission. During the management stage, only authorized users can join the multicast tree and obtain multicast session keys. Moreover, only a trusted switch can join in the multicast forwarding tree. Our scheme ensures that the node can verify the data from the other trusted nodes, which greatly improves its ability against pollution and eavesdropping attacks. During data transmission, the node only focuses on how to forward the data to maintain high performance of the network coding. To provide more flexible and efficient multicast services, we will further improve the system dynamics in the future, especially on how to deal with the frequent changes of members.

Acknowledgements. This work is supported by the National Basic Research Program of China (2012CB315901), and also supported by National Natural Science Foundation of China (61373006).

References

1. Winter, M.: Data center consolidation: A step towards infrastructure clouds. In Proceedings of the 1st International Conference on Cloud Computing. 190-199. (2009)
2. Chen, H. C., Yang, C. Y., Su, H. K., Wei, C. C., Lee, C. C.: A Secure E-Mail Protocol Using ID-based FNS Multicast Mechanism. Computer Science and Information Systems 11(3):1091C1112. (2011)
3. Li, D., Yu, J., Yu, J., Wu, J.: Exploring efficient and scalable multicast routing in future data center networks. Proceedings - IEEE INFOCOM, 34(17), 1368 - 1376. (2011)
4. Li, H., Li, P., Guo, S.: Efficient privacy-preserving multicast in cloud data centers. Communications (ICC), 2014 IEEE International Conference on. 810 C 815. (2014)
5. Zhang, X. C., Yang, M. H., Geng, G. G., Luo, W. M. Li, X. F.: A Two-Tiered Reliable Application Layer Multicast. ComSIS Vol. 8, No. 3. (2011)
6. Ahlswede, R. Cai, N., Li, S. Yeung, R.: Network information flow. IEEE Transactions on Information Theory, 46(4): 1204-1216. (2000)

7. Li, S. Y. R., Cai, N., Yeung, R. W.: On theory of linear network coding. In Proceedings. International Symposium on Information Theory. pp. 273C277.(2005)
8. Li, S. Y. R., Yeung, R. W., Cai, N.: Linear network coding. IEEE Trans. Inform. Theory, 49(2):371-381. (2003)
9. Han, K., Ho T., Koetter, R., Medard, M., Zhao, F.: On network coding for security. In IEEE MILCOM (2007).
10. Zhao, F.Kalker, T.Medard, M.Han, K. J.: Signatures for content distribution with network coding. IEEE International Symposium on Information Theory, NiceFrance 556-560. (2007)
11. Boneh, D., Freeman, D., Katz, J., Waters, B.: Signing a Linear Subspace: Signature Schemes for Network Coding. Lecture Notes in Computer Science, 68-87. (2008)
12. Yu, Z., Wei, Y., Ramkumar, B. Guan, Y.: An efficient scheme for securing XOR network coding against pollution attacks. in Proc. of the 28th IEEE International Conference on Computer Communications (INFOCOM). (2009)
13. Agrawal, S., Boneh, D.: Homomorphic MACs: MAC-based integrity for network coding. In Proceedings of ACNS 09, volume 5536 of LNCS: 292-305. (2009)
14. Agrawal, S., Dan, B., Boyen, X., Freeman, D. M.: Preventing Pollution Attacks in Multi-source Network Coding. Lecture Notes in Computer Science, 161-176. (2010)
15. Dong, J., Curtmola, R., Nita, R. C.: Practical Defenses Against Pollution Attacks in Intra-Flow Network Coding for Wireless Mesh Networks. in ACM WiSec09. 111-122. (2009)
16. Li, Y., Yao, H,, Chen, M., Jaggi, S., Rosen, A.: RIPPLE Authentication for Network Coding. Proceedings - IEEE INFOCOM, 29(16):2258-2266. (2010)
17. Perrig, A., Canetti, R., Tygar, J. D., Song, D.: The TESLA Broadcast Authentication Protocol. RSA CryptoBytes, vol. 5. (2002)
18. Wu, X., Xu, Y., Yuen, C., Xiang, L.: A Tag Encoding Scheme against Pollution Attack to Linear Network Coding. IEEE Transactions on Parallel and Distributed Systems, 25(1):33-42. (2014)
19. Ho, T., Medard, M., Koetter, R., Shi, J., Effros, M. Karger, D. R.: On randomized network coding. In: Proc. of the 41st Annual Allerton Conf. on Communication, Control, and Computing. (2003)
20. Kozat, U.C., Liang, G., K?kten, K.: On diagnosis of forwarding plane via static forwarding rules in software defined networks . In: Proc. of the IEEE INFOCOM. (2014)
21. Cohen, R., Eytan, L.L., Naor, J.S., Raz, D.: On the effect of forwarding table size on SDN network utilization. In: Proc. of the IEEE INFOCOM. (2014)
22. Lan, J. L.: Research on the architecture of the reconfigurable fundamental Information communication network. Report of the National Basic Research Program of China. (2012)
23. Liu, S., Hua, B.: NCoS: A framework for realizing network coding over software-defined network. 2014 IEEE 39th Conference on Local Computer Networks (LCN)IEEE Computer Society. 474-477. (2014)
24. Tan, X., Li, H., Zhu, Z., Yu, C., Qin, L.: LMTM: Multi-tree multicast with inter-layer network coding for layered multimedia streaming. Communications and Networking in China (CHINACOM), 2013 8th International ICST Conference on (Vol.12, pp.871-876). IEEE (2013)
25. Jadhav, J., Gadage, S.: Secure and fast data transfer with network cofing. International Journal of Computer Science Engineering and Information Technology Research (IJCSEITR) ISSN(P): 2249-6831; ISSN(E): 2249-7943. Vol. 3, Issue 5, 153-164. (2013)
26. Nagata, A., Tsukiji, Y., Tsuru, M.: Delivering a File by Multipath-Multicast on OpenFlow Networks. Intelligent Networking and Collaborative Systems (INCoS), 2013 5th International Conference on (pp.835-840). IEEE. (2013)
27. Shields, C., Garcia-Luna-Aceves, J. J.: Khip – a scalable protocol for secure multicast routing. Acm Sigcomm, routing. Acm Sigcomm, 29, 53-64. (1999)
28. Kong, L., Shen, H.: hieving Inter-domain Routing Security Based on Distributed Translator Trust Model. Computer Science and Information Systems 12(4):1327-1344. (2015)

29. Kreutz, D., Ramos, F. M. V., Verissimo, P.: Towards secure and dependable software-defined networks. Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networkingACM.55-60. (2013)
30. Yan, Z., Prehofer, C.: Autonomic Trust Management for a Component-Based Software System Dependable and Secure Computing IEEE Transactions on. 8(6):810-823. (2011)
31. Lantz, B., Heller, B., Mckeown, N.: A network in a laptop: rapid prototyping for software-defined networks. Acm Sigcomm Hotnets Workshop. (2010).
32. Jaggi, S., Sanders, P., Chou, P. A., Effros, M., Egner, S., Jain, K., Tolhuizen, L. M. G. M.: Polynomial time algorithms for multicast network code construction. Information Theory IEEE Transactions on, 51(6), 1973-1982. (2005)

Kaixiang Huang received his M.S. degree in Computer Application from Institute of Zhengzhou Information Technology, China, and is now a PhD candidate in computer application. He has worked in the area of broad band multicast. He is skilled in network security technology.

Yue Chen received his M.S. degree in Computer Software from Institute of Zhengzhou Information Technology, China and his Ph.D. degree in Communication and Information System from China National Digital Switching System Engineering and Technological Research Centre, china. His research area broad band multicast. He is currently with network security technology.

Hongyong Jia received his M.S. degree in Computer Application from Institute of Zhengzhou Information Technology, China, and his Ph.D. degree in Cryptography from Beijing University of Post and Telecommunication, China. He has worked in the area of applied cryptography and network security. He is proficient in developing data packet authentication methods for the cloud and complex network environment.

Julong Lan received his M.S. degree in Communication and Electronic System from Xidian University and his Ph.D. degree in Communication and Information System from China National Digital Switching System Engineering and Technological Research Centre. He has worked in the areas of network router, parallel switch structure and IPv6 technology. He is currently the Chief Scientist in Chinese national basic research project-Flexible Architecture of Reconfigurable Infrastructure.

Xincheng Yan received his B.A. degree in Software Engineering from Institute of Zhengzhou Information Technology, China, and is now a M.A. candidate in computer application. He has worked in the area of network security and he is skilled in access control of cloud data.

Zhiwei Wang received his Ph.D. degree in cryptography from the Beijing University of Posts and Telecommunications, Beijing in 2009. He is an associate professor in the department of information security at Nanjing University of Posts and Telecommunications. Currently, He serves as a research associate in the department of computer sciences at the University of Hongkong. His research interests include digital signatures, provable security, cryptographic protocols and cloud security.

Received: February 22, 2016; Accepted: May 24, 2016.