Cross-layer access control in publish/subscribe middleware over software-defined networks

Yang Zhang [a,*], Huiyu Zhou [b], Jun-liang Chen [a]

[a] State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China
[b] Department of Informatics, University of Leicester, United Kingdom

ARTICLE INFO

Keywords:
SDN
Publish/subscribe
Access control
Security

ABSTRACT

When technologies of software-defined networks (SDNs) provide a chance to improve the quality of service (QoS) of publish/subscribe middlewares, new chances are also arising for adversaries to attack the networks and the middlewares. We here propose a cross-layer access control solution to protect the publish/subscribe middleware over SDNs. Applications over a publish/subscribe middleware interact by an indirect, anonymous and multicast event communication paradigm, where we hope that the applications, the middleware, and the underlying network collaborate to realize the access control of reading/writing events. The key issue is how to use the flow matching capability of SDN switches to efficiently and securely enforce complex authorization policies that include multiple conjunction and disjunction structures. It is required to resist against the collusion attacks of SDN controllers and subscribers when the middleware/network is partially delegated to enforce the authorization policies of publishers. In our cross-layer solution, a policy representation method is presented to encode authorization policies into flow entries with high data compression and security, and a two-party computation method is presented to carry out secret sharing for defeating malicious SDN controllers and subscribers. Finally, our solution is evaluated to show its effectiveness.

1. Introduction

In Internet of Things (IoT) applications, publish/subscribe middlewares are needed to build a communication infrastructure for multiple consumers to access real-time and coherent sensor data, and software-defined networks (SDN) can be used to address the difficult issue of improving the quality of service for delivering events [1]. That is, in the existing publish/subscribe middlewares over traditional networks [2–4], a detour to brokers is required for matching events against the stored filters as well as additional communication processing, while, in the ones over SDNs, standards like OpenFlow [5] define the interfaces to directly install and change flow tables on SDN switches for direct and controllable event matching and forwarding.

The GridStat project [6] showed the importance of SDN technologies for the publish/subscribe middleware in IoT scenarios, where it developed a publish/subscribe-based communication infrastructure for smart grids, a subscriber can express to the infrastructure its interest in data measured by phasor measurement units, and the infrastructure can deliver to the subscriber in time all events matching the expressed interest. In the project, network routers were specially designed for realizing the QoS for event deliveries. But this special design limited its applications. In SDN networks, this limitation can be significantly reduced by the user-programmability of the underlying routers and switches.

Although SDN technologies provide chances to construct more efficient publish/subscribe middlewares as an IoT communication fabric, the openness and programmability of SDN networks also provide chances for adversaries to corrupt SDN controllers, forge flow tables, and exploit the vulnerability of northbound and southbound interfaces [7–10]. On the contrary, the SDN programmability also gives a new way to implement network/application security. How to protect a publish/subscribe middleware over SDNs thus becomes important and explorable. In this paper, we focus on the access control of reading/writing its events.

For the access control of events in a publish/subscribe middleware, the subscribers do not directly obtain events from the publishers, and the publishers cannot directly reject the subscribers to access to their events. Thus anonymous, indirect, and multicast communications bring up challenges for a publisher to control the access to its published events.

In the work of [11] and the publish/subscribe security standard proposed by OMG [12], the publish/subscribe middleware together with their clients (publishers and subscribers) was organized into multiple topic domains over the traditional networks. In each domain, they had their own security management servers and home brokers. Clients delegated their authorization functions to the home brokers and the target domain's security management servers. Although it addressed

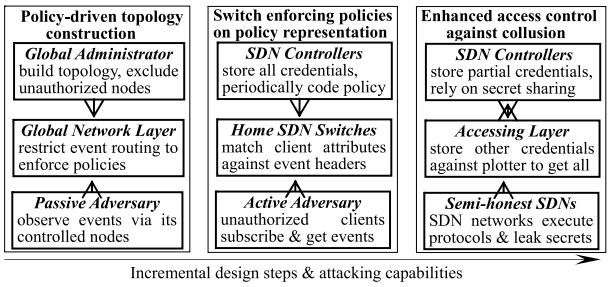


Fig. 1. Roadmap.

the access control issue in publish/subscribe middlewares, such over-delegation in some degree violated the security requirements of clients controlling their events' accessing [13,14]. In the work of [13,15], event encryption-based schemes were proposed to avoid the delegation of access control. Such over-user-controlled schemes however lost scalability and efficiency [11,12].

In our work, we also adopt the access control framework having multiple topic domains, but we do not require the clients to completely delegate their access control functions to the brokers and the target domain servers. That is to say, we adopt a cross-layer approach to accomplish the access control, where SDN controllers are used to manage some security tokens for securely encoding authorization policies into flow tables, SDN switches enforce the policies by their flow matching capabilities, and publishers and subscribers still control authentication, authorization, and their credential disseminations.

In the above discussion, the major challenge is to high-compressed securely encode complex authorization policies (e.g. including multiple conjunction and disjunction structures) into flow tables, and use the flow matching capability to efficiently enforce authorization policies, when we try to provide access control in a publish/subscribe middleware over SDN networks. The enforcement of authorization policies in the traditional publish/subscribe brokers (powerful servers) cannot be directly carried out in SDN networks, because SDN switches have no capabilities to process high-level languages and complex data structures, and the policy enforcement on SDN controllers will evidently delay event forwarding with losing the advantages of SDNs in building publish/subscribe middlewares, *i.e., detouring routing paths and heavily burdening SDN controllers*. It requires using a little flow table space and a limited part of event headers to carry out the access control.

The roadmap of our incrementality method, *illustrated in* Fig. 1, is to design an access control solution against a powerful adversary by incrementally increasing the adversary's attacking capabilities, where the SDN network is decomposed into multiple partitions with each partition as a publish/subscribe node (*or as a network node in different contexts*), each controller is in charge of one partition with multiple SDN switches, a global administrator manages all local SDN controllers, and the accessing layer is deployed on clients for their locally reading/writing events in the publish/subscribe middleware. In this paper, our contributions are below.

(1) A global administrator of the SDN network carries out a topology construction for different event topics, which restricts unauthorized publish/subscribe nodes to visit the events, *i.e., the events being disseminated among authorized nodes as far as possible.* In this policy-driven topology construction scheme, a publish/subscribe node may be authorized for one event topic while it also may be unauthorized for the other event topic. The constructed topology for all the event topics is to maximize the number of common nodes among these corresponding event streams. This policy-driven topology will enhance the flow-table-based policy enforcement at the global network layer rather than on individual nodes connected to the subscribers. The event routing for one topic can also be rapidly computed over the unified topology for all the event topics [16–18].

(2) The matching capabilities of flow tables of SDN switches are used to realize the matching between an authorization policy and subscribers' attributes, where the policy has strong expressiveness (e.g. including attribute conjunction structures, attribute disjunction structures, and their mixtures), and SDN controllers securely encode these structures into a matching field of flow entries with high data compression. A two-party computation scheme is proposed to get a sharing secret for encoding, which forms a framework to resist against the collusion attacks of SDN controllers and subscribers. These two schemes together with the topology construction scheme compose a cross-layer access control solution for the publish/subscribe middleware over SDN networks.

(3) A policy representation method is presented to build the cornerstone for the above solution, which represents an authorization policy having complex structures by a non-structure plain bit string, and is also proved to be secure.

The remainder of this paper is structured as follows. Section 2 describes the related work. Section 3 describes the preliminaries. In Section 4, the policy-driven topology construction scheme is described. In Section 5, the access control based on the matching capabilities of flow tables of SDN switches is described. In Section 6, we present the performance evaluation of our solution. Finally, conclusions are given in Section 7.

2. Related work

There are several famous topic-based publish/subscribe systems, *e.g., SCRIBE* [19], *Bayeux* [20], *TERA* [21], *Corona* [22] and *NICE* [23]. These systems are often established based on an overlay topology over underlying traditional networks, and underlying routers cannot be controlled and scheduled by users such that their QoS and security is scanty in IoT scenarios.

Recent SDN technologies provide a high potential to improve the quality of services of publish/subscribe middlewares. The approaches shown in [24–28] presented examples of publish/subscribe middlewares over SDNs. They discussed event filtering and routing over SDNs, but the access control requirements of these middlewares were not comprehensively explored.

The security problems in publish/subscribe middlewares involve participant privacy [29–32], event confidentiality [12,13,15], and access control [11–13,15]. The work of [32] tried to protect subscription privacy in a publish/subscribe system. The problem of protecting publication privacy was considered in the work reported in [14,29,31]. They did not consider supporting access control capabilities.

Based on CP-ABE (Ciphertext-Policy Attribute-based Encryption) [33], the work of [15] presented an access control scheme for events delivery in a brokerless publish/subscribe system. In the scheme, events were encrypted by a symmetric encryption operation and the symmetric key was encrypted by CP-ABE. The work of [13] combined an attribute-based encryption scheme and a multi-user searchable data encryption scheme to construct an access control solution for publish/subscribe middlewares, where brokers matched subscriptions without knowing the encrypted sensitive data. The DDS security standard [12] defines an access control framework DDS-AC for publish/subscribe systems, where a publisher is allowed to set up topic domains, and domain servers are delegated to take in charge of authorization. In our work, we also adopt this multiple-domains framework, but publishers do not need to delegate their authentication and authorization to the domain servers. That is to say, only authorization enforcement is undertaken by the publish/subscribe middleware, and others are controlled by the publishers. In addition, how to use the matching capabilities of SDN switches to realize the enactment of authorization policies was not considered in these existing work.

In [7–10], the security problems of SDN network were discussed, including corrupting SDN controllers, forging flow tables, and exploiting the vulnerability of northbound and southbound interfaces. But they did not provide a full-fledged security mechanism to directly protect the

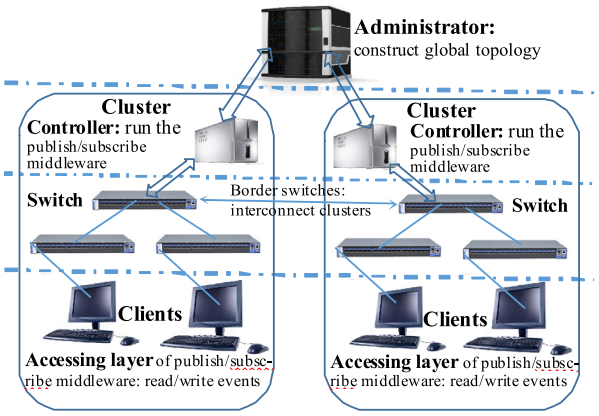


Fig. 2. Our publish/subscribe middleware over SDNs.

publish/subscribe middleware over SDNs. In our work, the controller encodes the policies and subscribers' attributes into the flow tables when the authorization policies are modified; and the SDN switches enforce authorization by flow table matching.

The work of [34–36] presented some access control methods for SDN networks, but they only focused on protecting SDN controllers and flow tables rather than on enforcing authorization policies in SDN switches for publish/subscribe middlewares. The work of [37] also advocated using SDN switches to enforce security policies, and proposed a guaranteed update scheme of security policies to install transferred policies into multiple SDN switches. The work of [38] adopted the idea of [37], and addressed the issue of detecting policy conflicts and resolving the conflicts in SDN networks. In our work, we believe that the encoding of security policies is more important than only deploying them on SDN switches because it can largely reduce the size of policies and provide the security guarantee for transferring authorization policies into flow entries.

3. Preliminaries

3.1. Publish/subscribe middleware over SDNs

In our publish/subscribe middleware [28], *illustrated in* Fig. 2, the SDN network is partitioned into multiple partitions (*a partition is also called a broker, a node or a cluster)*. Each cluster is managed by a local controller as an interconnected network zone, which consists of a set of SDN-configurable switches and clients. A pair of border SDN switches connects two neighboring clusters. We follow the topic-based subscription model [16,17,19], *i.e., an event being composed of a topic name and a set of attribute–value pairs*. Multiple topics form a topic tree.

In Fig. 2, our middleware consists of three layers:

(1) There is a global administrator to manage all the local controllers to form a three-layer architecture: *global administrator, clusters networking, and local accessing layers*. The global administrator runs on a server to create and store event topic trees, event schemes, security policies, settings configuring, and performance strategies. The key function is to construct a topology for event streams in the middleware.

(2) In the clusters networking layer, SDN controllers maintain the clusters, update the link states, advertise subscriptions, compute event routes, and install flow tables on SDN switches. SDN switches forward event flows by the installed flow tables. The key function of controllers is to store authorization credentials and encode policies and attributes into flow entries on SDN switches.

(3) The local accessing layer runs on clients including publishers and subscribers, which provides the clients local interfaces to read and write events, *i.e., locally accessing to the publish/subscribe middleware*.

Each flow can define any content used to match against the event header fields, such as VLAN tags or IP address [39,40]. In this paper, we assume that the publish/subscribe middleware will manage flows with their matching fields corresponding to IP-Multicast addresses [28]. An IPv6 destination address is used to encode the topic name, event priority, and authorization policy, illustrated in Fig. 3. How to encode topic trees can be referred to our previous work [28].

If the IP destination address of events is not matched against the policy field of the flow entries, the events will be discarded; otherwise the events are forwarded to the switch's specific ports that an authorized subscriber/node is connected to.

In this paper, we only assume that our underlying SDN network is semi-honest, *honestly executing given protocols but arbitrarily leaking sensitive information in the protocols*, without assuming a special secure SDN.

3.2. Attribute-based policy representation

In this paper, we adopt an attribute-based access control model [41], because peer-to-peer attribute matching is appropriate for the publish/subscribe middleware with decoupling interaction paradigms. An authorization policy for events is made based on attributes. We then use an access tree to represent an attribute-based authorization policy and define what attributes satisfy the policies.

Access Structure. *Let* $\{S_1, S_2, \ldots, S_n\}$ *be a set of subjects. A collection* $\Lambda \subseteq 2^{\{S_1, S_2, \ldots, S_n\}}$ *is monotone, when, for any two sets* B, C, $B \in \Lambda$, $B \subseteq C$ *results in* $C \in \Lambda$. *A monotone access structure* Ω *will be a monotone collection of non-empty subsets in* $\Lambda \subseteq 2^{\{S_1, S_2, \ldots, S_n\}}$. *The sets belonging to* Ω *will be called the authorized sets, and the sets that are not in* Ω *will be called the unauthorized sets.*

Access Tree. *We assume* T *is an AND/OR tree which is an access structure. Each non-leaf node of* T *will be a threshold gate that is specified by a threshold value and its children. The number of the children of a node* x *is denoted by* num_x, *and the threshold value is denoted by* k_x *with* $0 < k_x \leq num_x$. *If* $k_x = 1$, *then the threshold gate is an OR gate. If* $k_x = num_x$, *then the threshold gate is an AND gate. Each leaf node of* T *will be specified by an attribute and a threshold value* $k_x = 1$.

The parent of node x in T will be denoted by $parent(x)$, and its attribute will be denoted by $attr(x)$ if it is a leaf node. All the children of a node can be ordered, and the order number of x will be denoted by $index(x)$.

Satisfying an access tree. *We assume that* T *is an AND/OR tree with a root* r. T_x *is a sub-tree of* T *rooted at* x. *When a set of attributes* γ *will satisfy the access tree* T_x, *it will be denoted by* $T_x(\gamma) = 1$. *For a leaf node* x *of* T, $T_x(\gamma) = 1$ *when* $attr(x) \in \gamma$. *For a non-leaf node* x *of* T, $T_x(\gamma) = 1$ *when its at least* k_x *children will return 1 during the evaluation on whether* $T_{x'}(\gamma) = 1$ *for its each child* x'.

In order to encode an authorization policy into a secret policy representation, *i.e., one non-structure plain bit string*, we model this policy representation scheme: making an authorization policy by an access tree, *i.e., an initialization algorithm*; issuing authorization credentials according to the users' attributes and the access tree, *i.e., an authorization algorithm;* generating a secret policy representation, *i.e., an algorithm generating the representation;* and reconstructing the secret policy representation by the users' authorization credentials, *i.e., an algorithm reconstructing the representation*; where the policy matching means that the policy representation reconstructed by the users is equal to the policy representation from the issuer, *i.e., being authorized.*

Definition 1 (*Attribute-based Policy Representation Model: AP*^{2}*M*).

1. **Setup.** For the universe attributes $\mu = \{1, 2, \ldots, n\}$, the algorithm generates a private scheme parameter $prPar$ and a public parameter $pbPar$.
2. **Initialization.** For an event topic tp, the publisher i generates an access tree $T_{i,tp}$ as the authorization policy, and the scheme parameters, where an actual secret $\alpha_{i,tp}$ is also generated with respect to the access tree.

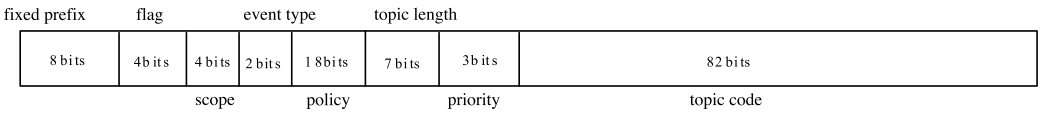


Fig. 3. IPv6 destination address embedding topic name and policy.

3. **Authorization**. Given a user's attributes γ, the publisher i generates a set of authorization credentials $creS$, if γ satisfies $T_{i,tp}$. $creS$ can be split to two parts: one for topic, the other for the user.
4. **GeneratingRepresentation**. Given the access tree $T_{i,tp}$ and a time period, the publisher i chooses a random number, and generates a secret policy representation $dCre$ and a public parameter par with respect to the presentation, called a representation parameter.
5. **ReconstructingRepresentation**. Given the public parameter $pbPar$ and the presentation parameter par, users can reconstruct the secret policy representation $dCre$ according to their authorization credentials $creS$.

Definition 2 (*Selective Attribute Set* ***Security Game*** *[41]*).
Init. The adversary chooses the set of attributes, γ, that he will be challenged on.
Setup. The challenger invokes the Setup algorithm in Definition 1 to generate public parameters sent to the adversary.
Phase 1. The adversary launches queries for secrets for access structures that γ does not belong to.
Challenge. The adversary will submit two time periods m_0, m_1. The challenger will flip a coin b, and then uses m_b with γ. The policy representation will be given to the adversary.
Phase 2. The adversary launches queries for secrets for access structures that γ does not belong to.
Guess. The adversary will guess b and return a bit.

4. Policy-driven publish/subscribe topology for multiple topics

The work of [16] reported a method to construct the overlay topology for multiple topics in a publish/subscribe middleware. The constructed topology is trustful for publishers to disseminate confidential information. The topology construction of publish/subscribe middlewares over SDNs, however, has different requirements for access control, as follows:

(1) From the perspective of the physical topology of SDN networks, sensitive events may be disseminated through the publish/subscribe nodes which do not have rights to read them [16–18,42], although they pass through the trusted nodes in an overlay topology. It is desirable that, from all the perspectives, the number of the unauthorized nodes be minimal in a constructed topology that connects together all the subscribers and publishers.
(2) One node may cross multiple event streams with different authorizations: authorized in some streams but unauthorized in the others. How to use extra nodes outside all the streams to form connected sub-graphs is important. A criterion is to maximize common node numbers among streams without violating any policy. Anyway, the event routing cost should also be minimized.

4.1. Problem statement

We try to construct a connected topology that unauthorized nodes are excluded so that events are disseminated without being visited by adversaries. In addition, the event delivery cost is also minimal on the constructed topology. We then formally define it.

We assume that $G = (V, E)$ is an undirected graph representing the SDN network, where $V = \{v_i | 1 \le i \le n\}$ is considered as the set of n nodes, and $E = \{e_j | 1 \le j \le m\}$ is considered as the set of m communication links connecting them. Let c_j be the communication cost of link e_j.

Assume there exist r event streams for r topics that are subscribed by some of the nodes in V. We use $W = \{w_k | 1 \le k \le r\}$ to denote the set of r event streams. Let S_k be the set of the publisher nodes that publish w_k and D_k for the set of nodes that subscribe the event stream w_k. We use $|S_k|$ and $|D_k|$ to denote the cardinality of S_k and D_k, respectively, where $|S_k| \ge 1$, $|D_k| \ge 1$, $1 \le k \le r$. Let $\delta_k = (w_k, S_k, D_k)$ denote as the multicast of w_k, and $cost(\delta_k)$ as the total communication cost of the multicast.

Definition 3 (*AC-TC-SDN*). The construction problem of a connected topology for topics to satisfy access control requirements in a publish/subscribe middleware over SDN networks, called AC-TC-SDN, is defined as follows:

Given a SDN network $G = (V, E)$, a set of event streams $W = \{w_k | 1 \le k \le r\}$ with multicasts $\delta_k = (w_k, S_k, D_k)$, $1 \le k \le r$, the goal of AC-TC-SDN is to find a series of connected sub-graphs (or one sub-graph) $F_k = (V_k = \{v_{k,i} | 1 \le i \le n_k\}, E_k = \{e_{k,j} | 1 \le j \le m_k\}), 1 \le k \le r$ with $c_{k,i}$ denoting the cost of a link $e_{k,i} \in E_k$, for each multicast δ_k, such that:

(1) $\text{minimize}(|\bigcup_{k=1}^{r}(V_k - S_k \cup D_k)|)$;

(2) $\text{minimize}(\sum_{k=1}^{r} \text{cost}(F_k) | \text{cost}(F_k) = \sum_{j=1}^{m_k} c_{k,j})$.

The problem of AC-TC-SDN is to minimize the node number out of all the event stream multicasts, and the connection cost of the constructed topology.

Theorem 1. *AC-TC-SDN is NP-hard.*

Proof. For the goal: $\text{minimize}(|\bigcup_{k=1}^{r}(V_k - S_k \cup D_k)|)$, we can assign each link by the same cost such as 1, in the event stream δ_k, when the link is not $(v_{k,i}, v_{k,j})$, $v_{k,i}, v_{k,j} \in (S_k \cup D_k)$, and the cost on other links is less than $1/(|V| - 1)$.

Then, finding the minimal number of nodes outside of $(S_k \cup D_k)$ to connect all the nodes in $(S_k \cup D_k)$ is equal to finding a Steiner tree [43] for $(S_k \cup D_k)$ in G.

According to [44,45], the Steiner tree problem is NP-hard. Then AC-TCO-SDN is NP-hard when multiple Steiner trees should be found in G with maximizing their common parts. In addition, minimizing the communication cost further complicates the computation.

4.2. Solving AC-TC-SDN

From the perspective of the physical topology of SDN networks, SDN network nodes are classified into three groups: authorized nodes, unauthorized nodes, and non-determinative nodes. An event stream $\delta_k = (w_k, S_k, D_k)$ also represents the authorization for event topic $topic_k$, *i.e., all the nodes in* D_k *having rights to read events with* $topic_k$, published by S_k. For all the nodes that have no rights to read events with $topic_k$, we use $\overline{V}(S_k)$ to denote the node set. $\overline{V}(s)$, $s \in V$ indicates the nodes that have no rights to read the events published by the node s. $V - (S_k \cup D_k) - \overline{V}(S_k)$ indicates the non-determinative node set that is not explicitly authorized or unauthorized. A non-determinative node can be used as an extra node to connect S_k, D_k.

In order to reduce extra nodes in a multicast tree for the event stream, the links between the non-publishers and the non-subscribers are attached by a large cost, such as the production of the maximal cost on all the edges and the maximal length of the paths in the graph. In this case, the shortest path will be computed by preferentially selecting the links between the publishers and subscribers.

For the SDN network $G = (V, E)$, the maximal link cost is denoted by c_{max} and the maximal length of paths is denoted by len_{max} such as $|V| - 1$. For the event stream $\delta_k = (w_k, S_k, D_k)$, we then compute the shortest paths between nodes in $S_k \cup D_k$, where $D[s, d]$ is used to denote the cost of the path from a node s to d. The cost $D[v_i, v_j]$ from a node $v_i \in V$ to another node $v_j \in V$ is initialized-$Initialization(G, S_k, D_k, \{D[v_i, v_j] | v_i, v_j \in V\})$:

$$D[v_i, v_j] = \begin{cases} c_{i,j}, \ \exists e_{ij}, \ v_i, v_j \in (S_k \cup D_k) \\ c_{i,j} \leftarrow len_{max} * c_{max} + c_{i,j}, \exists e_{ij}, \\ (v_i \notin (S_k \cup D_k) \ or \ v_j \notin (S_k \cup D_k)) \\ 0, \ if \ i = j \\ \infty, \ otherwise. \end{cases}$$

After the shortest paths between S and D have been computed with links between nodes in $S \cup D$ being preferentially selected whilst unauthorized nodes being excluded, the fast algorithm for Steiner trees [46] can be used to compute the expected topology, as in Algorithm 2.

Event routing paths can be rapidly built on the constructed topology such as extracting one event stream tree from the multicast forest. The constructed topology will maximize the number of common nodes among all the event streams besides adding extra nodes.

This topology construction scheme is carried out on the global server/administrator of SDNs, which costs at most the time of $O(|\sum_{k=1}^{r} S_k \cup D_k| |V|^2)$. The communication optimization of our method can be discussed based on the fast algorithm shown in [46].

To avoid the over-usage of one link, some integer programming algorithms [47] can be used to compute the shortest path, *i.e., revising Line 5 in Algorithm 1*.

Algorithm 1. $ShortPath(G, S, D)$ **from** S ***to*** D

Input: $G = (V, E)$ *with costs on edges,* $\delta = (w, S, D)$

Output: *The shortest paths* $\{D[s,d] | s,d \in S \cup D\}$

1. *Initialization* $(G, S, D, \{D[v_i, v_j] \mid v_i, v_j \in V\})$
2. **foreach** *node* $s \in S \cup D$,
3. $tpS = \{s\}$, $tpD = V - tpS$
4. **repeat until** $(S \cup D) \subseteq tpS$
5. $D[s,j] = min\{D[s,v_j], D[s,v_x] + c_{x,j} | v_x \in tpS, \ s \in S \cup D, \ v_x \in V - \overline{V}(s), \ \exists e_{x,j}, v_j \in tpD - \overline{V}(s)\}$

   //explicitly unauthorized nodes are not used
6. $tpS = tpS \cup \{v_j\}, tpD = tpD - \{v_j\}$,
7. **continue**
8. **return** *all the shortest paths* $\{D[s,d] | s,d \in S \cup D\}$

Algorithm 2. Multicast Forest for W

Input: $G = (V, E)$ *with cost on edges,* $\{\delta_k | 1 \le k \le r\}$

Output: *A multicast forest* $STree$, *or failed*

1. **let** $S = \bigcup_{k=}^{r} S_k$, $D = \bigcup_{k=}^{r} D_k$,
2. $\{D[s,d] | s,d \in S \cup D\} = ShortPath(G, S, D)$
3. **for** $S \cup D$ *with* $\{D[s,d] | s,d \in S \cup D\}$ **do**
4. **compute** *a minimal spanning graph* G', *or several minimal spanning graphs* $G_1, \cdots, G_{r'}$

   // according to the fast algorithm in [33]
5. **foreach** $\delta_k, 1 \le k \le r$ **do**
6. **if** $((S_k \cup D_k) \cap G_i \neq \emptyset \wedge (S_k \cup D_k) \cap G_j \neq \emptyset) \vee ((S_k \cup D_k) \not\subset (\bigcup_{x=1}^{r'} G_x \cup G'))$, $1 \le i \ne j \le r'$
7. **return** *failed* // δ_k *has no connected topology*
8. **compute** *a Steiner tree* ST_k *on* G' *or* $G_i, 1 \le i \le r'$ *to span* $S_k \cup D_k$ *// according to the fast algorithm in [33]*
9. **return** $STree = ST_1 \cup \cdots \cup ST_r$

5. Access control based on SDNs' matching capabilities

Subscribers receive events from the publish/subscribe middleware, and do not directly access to the publishers. A publisher does not know who reads its published events. The access control of the events published by the publishers will be anonymous and indirect. In our solution, the publishers will produce authorization policies and issue authorization credentials to the subscribers; and the middleware will decide whether or not it delivers the events to the subscribers according to the publisher's authorization policies, *i.e., enforcing the authorization*. Accesses to the middleware resources such as registering to the network will be managed by the middleware, which can be combined with the one of reading/writing events, and is not further discussed in this paper.

Our flow matching based access control consists of two schemes: policy matching and resisting against collusion attacks. In the policy matching scheme, the matching between authorization policies and users' attributes is assumed by the matching between the flow entries in SDN switches and event headers, *i.e., the users' attributes corresponding to flow entries in the SDN switches connected to the users*. In the scheme of resisting against collusions, a two-party secret sharing is carried out to secretly encode authorization policies into the flow entries for defeating malicious SDN controllers and subscribers.

Each subscriber is connected to its local switch and controller, which are treated as the home broker of delivering events, *regarding the local SDN controller as a home controller, and the SDN switch as a home switch*.

5.1. Policy matching by flow tables

The policy matching scheme is illustrated in Fig. 4, and the involved notations are listed in Table 1. A subscriber j registers to its home controller in order to access to the resources of the publish/subscribe middleware, and authenticates to a publisher i to obtain authorization credentials to read events with topic tp. The controller stores these authorization credentials of j, issued by the publisher i. The controller can use these authorization credentials and public parameters from the publisher i to compute the secret policy representation, *encoded into the policy field of flow entries shown in* Fig. 3 *as users*.

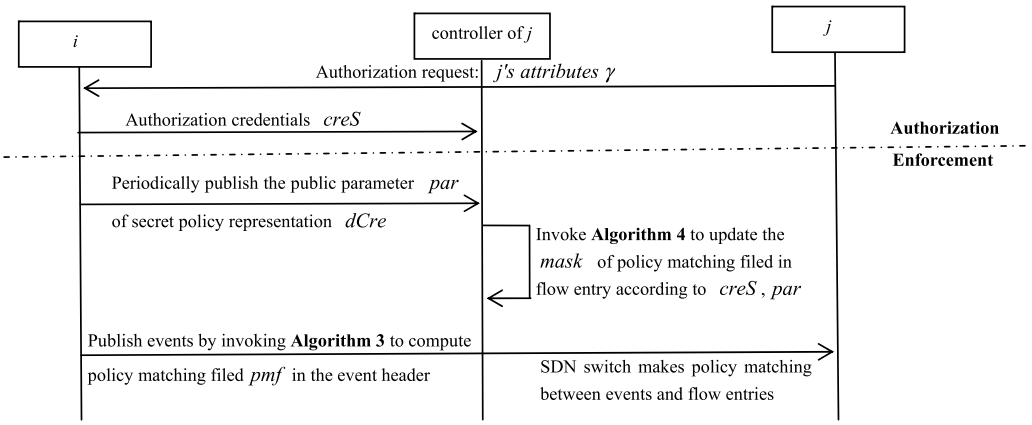


Fig. 4. Policy matching.

Table 1
Notations.

Notation	Meaning	Notation	Meaning
$\alpha_{i,tp}$	Secret value for tp	$creS$	Authorization credentials
$T_{i,tp}$	An access tree for tp	$dCre$	Secret policy representation
γ	Subscriber attributes	par	Public parameter of $dCre$
$hash$	Hash function	$mask$	The mask of matching field

Fig. 4 includes two stages: authorization stage and policy enforcement stage. At the authorization stage, a subscriber j requests the publisher to authorize reading events with topic tp. For the event topic tp attached by an access tree $T_{i,tp}$, the publisher i invokes the ***Initialization*** algorithm in **AP^{2}M** (shown in Section 3) to initialize parameters and contexts. When it gives authorization to the subscriber j according to j's attributes γ, the publisher i invokes the ***Authorization*** algorithm in **AP^{2}M** to issue authorization credentials to j: $creS$, which are stored in the home controller of j.

At the policy enforcement stage, the publisher i invokes the ***GeneratingRepresentation*** algorithm in **AP^{2}M**, which outputs the secret policy representation $dCre$ for this new time period, and the representation parameter par with respect to $dCre$. The publisher i will publish par, which is stored and used by the home controller of j to compute the secret policy representation $dCre$. That is to say, the publisher i directly generates $dCre$, and the home controller of j computes the same $dCre$ according to the authorization credentials $creS$ and the public parameter par. The "secret" word in the term of secret policy representation means that the policy representation $dCre$ is only known by the publisher i and the home controller of j (we further protect the policy representation in the next section to resist against malicious controllers).

The controller will update the mask of the policy matching field in the flow entries of the home SDN switch (shown in Fig. 3) according to **Algorithm 4**. The mask update of the flow entries is periodically performed, while the events are published on demand. When it publishes an event with topic tp, the publisher i invokes **Algorithm 3** to compute the policy matching field in the event header. **Statement 1** indicates that the home SDN switch can correctly deliver events with topic tp to the subscriber j if the attributes of j satisfy the access tree of i, *i.e., j having been authorized.*

Algorithm 3. Computing Policy Matching Field in Event Header by Publisher

Input. *The **Policy Matching Field** pmf' in the last period, and the secret policy representation $dCre$ in this period*

Output. *The **Policy Matching Field** pmf in this period*

1. $tpm = hash(dCre)$

2. $pmf = pmf' \otimes tmp$, $pmf' = tmp$

// $\otimes$ means bitwise AND.

3. return pmf

Algorithm 4. Computing the Mask of Policy Matching Field in Flow Entry by Controller

Input. *Authorization credentials $creS$, and public parameters par of secret policy representation in this period*

Output. *The **Mask of Policy Matching Field** $mask$*

1. $dCre = ReconstructingRepresentation(creS,par)$

//invoke **ReconstructingRepresentation** in **AP^{2}M**

2. $mask = hash(dCre)$

3. return $mask$

Statement 1. If the policy matching field fep in the flow entry and the last policy matching field pmf' in the event headers are ones (such as "1111111111") in the first period, then the events can be matched against the flow entry for an authorized subscriber.

Proof. In the first period, the masked policy matching field in the flow entry is $fep \otimes mask = hash(dCre)$, and the policy matching field in event headers is $pmf' \otimes tmp = hash(dCre)$. That is to say, they are matched. In addition, fep and pmf' are updated with $fep = mask = hash(dCre)$ and $pmf' = tmp = hash(dCre)$, i.e., $fep = pmf'$.

In other periods, they are matched if **ReconstructingRepresentation** in **AP^{2}M** can correctly compute the secret policy representation, where $pmf' \otimes tmp = pmf' \otimes hash(dCre)$, $mask \otimes fep = hash(dCre) \otimes fep$, and $fep = pmf'$, i.e., the masked policy matching field in the flow entry is equal to the policy matching field in the event headers.

$\Xi.A:$ **Home controller** Con_j **of** j

1. $dCre.tp = Reconstructing\ Representation(creS.\{D_x\}, par)$

   $cipher_{tp} = encrypt(dCre.tp)$

2. $cipher_{tp} \overset{send}{\rightarrow} j$

3. $hash_{sub} \overset{receive}{\leftarrow} j$

4. $Hash(dCre) = Hash(dCre.tp) \cdot hash_{sub}$

Fig. 5. The part A of protocol Ξ.

$\Xi.B:$ **Subscriber** j **-** ***its accessing layer***

1. $dCre.subscriber =$

   $Reconstructing\ Representation(creS.\{D'_y\}, par)$

   $hash_{sub} = Hash(dCre.subscriber)$

2. $hash_{sub} \overset{send}{\rightarrow} Con_j$

3. $cipher_{tp} \overset{receive}{\leftarrow} Con_j$

4. $encrypt(dCre) = cipher_{tp} \cdot encrypt(dCre.subscriber)$

Fig. 6. The part B of protocol Ξ.

5.2. Resisting against collusion attacks

The publisher may not completely trust the publish/subscribe middleware and the SDN network, and does not give all the authorization credentials to controllers, *i.e., against the malicious controllers*. In addition, it may also expect to resist against eavesdropping events. That is, we should enhance the above scheme, and the enhanced scheme will work together with a confidentiality mechanism to provide the end-to-end encryption of events.

In order to enhance the above scheme, we split authorization credentials into two parts: one part corresponding to a topic and the other part corresponding to a subscriber. The former is still stored at the home controllers, which can be used to compute one part of the secret policy representation. The latter is stored at the accessing layer of the subscriber, *i.e., being transparent to subscribers*, which can be used to compute the other part of the secret policy representation. The subscriber and its home controller can collaborate to compute the policy matching field in SDN switches without disclosing the secret policy representation.

For achieving the above objectives, we use a two-party protocol between a subscriber (including its accessing layer for cryptographic operations) and its home controller to secretly compute the policy matching field. We adopt a homomorphic encryption scheme as the basic confidentiality mechanism [48–50], and a homomorphic hash function [51–54] is used as the two-party computation operation. The two-party computation protocol Ξ of a secret is illustrated in Figs. 5–6, which includes a part A shown in Fig. 5, and the other part B shown in Fig. 6.

In the protocol Ξ, the home controller computes its part $dCre.tp$ of the secret policy representation according to its stored authorization credentials ($creS.\{D_x\}$) and the public parameter par. $dCre.tp$ is then encrypted: $encrypt(dCre.tp)$ under the public key of the publisher, and $encrypt(dCre.tp)$ is sent to the subscriber's accessing layer.

The subscriber's accessing layer computes its part $dCre.subcriber$ of the secret policy representation according to its stored authorization credentials ($creS.\{D'_y\}$) and the public parameter par. $dCre.subcriber$ is hashed: $hash(dCre.subcriber)$, and $hash(dCre.subcriber)$ is sent to the home controller.

There are two equations in the protocol:

$$encrypt(dCre) = encrypt(dCre.tp) \cdot encrypt(dCre.subscriber)$$
$$and\ hash(dCre) = hash(dCre.tp) \cdot hash(dCre.subcriber).$$

In the above two-party computation protocol, there is an implicit authentication channel, which is underlying to identify the controller and the subscriber. When it is encrypted or hashed, the policy representation is not disclosed to the subscriber's accessing layer and the home controller.

Given this secret sharing mechanism provided by the two-party protocol, we can enhance the policy matching scheme in the above section to resist against collusions and to provide confidentiality. It is illustrated in Fig. 7.

In Fig. 7, authorization credentials are split into two parts during the authorization stage. The two-party protocol is carried out periodically, which provides one secret policy representation in one period. During the enforcement stage, the publisher embeds the secret policy representation $dCre$ into the event's ciphertext $c = encrypt(e)$ by homomorphic addition, *i.e.,* $c = encrypt(e + dCre)$, where $encrypt()$ is a homomorphic encryption function.

The local accessing layer of the publish/subscriber middleware carries out the homomorphic operations to remove the embedded secret policy representation by its stored $encrypt(dCre)$ with ciphertext converting. That is, $c' = c - encrypt(dCre) = encrypt(e + dCre) - encrypt(dCre) = encrypt(e)$. c' is delivered to the subscriber.

The subscriber uses its private key to decrypt c' to recover the event e. The policy matching between the event header and the flow entries is carried out as in the policy matching scheme in the above section.

Although there are similarities between ours and that of [50], the differences are:

(1) The access tree in our method provides more expressiveness for authorization policies and their matching, while the work of [50] only allowed linear policy structure, *i.e., attribute conjunctions without disjunctions and their mixtures*.

(2) The access control functions in our method are separated from the encryption scheme with the ability to integrate them together on demand (e.g. flow-based authorization enforcement on switches working with independent encryption on clients in Fig. 7), which provides flexibility for deploying security mechanisms.

(3) The policy matching in our method is carried out by SDN switches, *i.e., making matching between flow entries and event headers as the policy matching.*

(4) The key difference is that our solution can resist against the collusions of the corrupted subscribers and home brokers, i.e., unable to get events sent to uncorrupted subscribers by using the colluded credentials.

5.3. Policy representation scheme

The work of [41] presented an attribute-based encryption scheme $\Re$ by access trees, but it cannot be directly used in our work because the policy representation scheme has different requirements:

(1) It should support splitting authorization credentials and secret sharing (discussed in Section 5.2), which is not considered and supported by $\Re$.

(2) It is not desired that the publish/subscribe middleware always carry out expensive encryption operations on each event for enforcing authorization, and the matching capabilities of SDN switches be used to efficiently match authorization policies against the subscribers' attributes. That is, we should represent a complex authorization policy as a short bit string in the flow tables and event headers, which is not computed for each event but periodically or policy-change-driven.

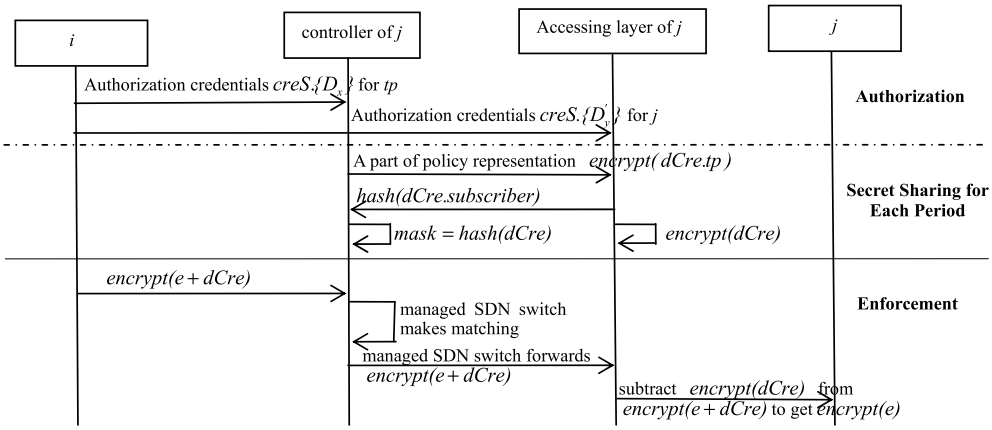
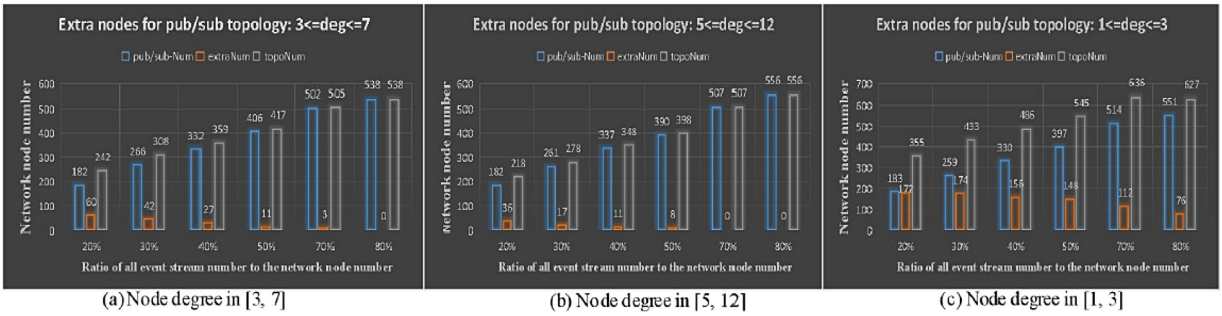
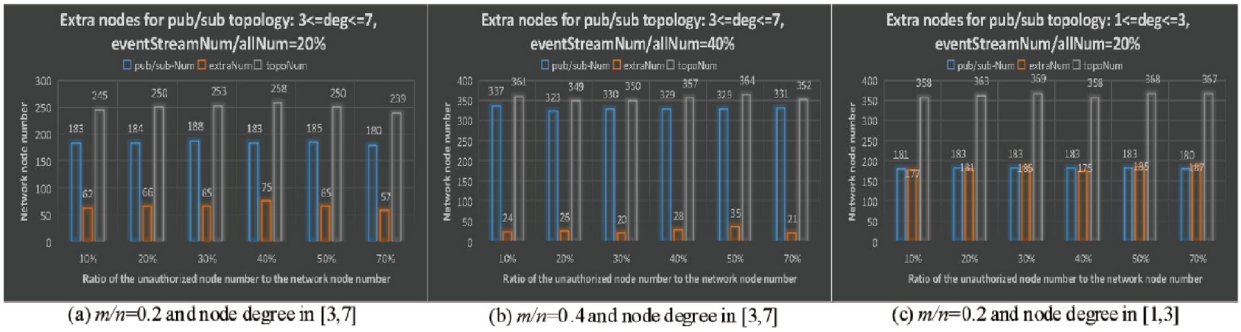


Fig. 7. Access control solution.



(a) Node degree in [3, 7] (b) Node degree in [5, 12] (c) Node degree in [1, 3]

Fig. 8. Extra nodes for different node degrees.



(a) m/n=0.2 and node degree in [3,7] (b) m/n=0.4 and node degree in [3,7] (c) m/n=0.2 and node degree in [1,3]

Fig. 9. Unauthorized node affection at different publisher/subscriber percent m/n and different node degrees.

(3) When event confidentiality is needed, it is desired that any encryption schemes be combined with the policy representation scheme, and encryption operations be carried out by the clients.

The complete description of the attribute-based policy representation scheme Π is presented in Appendix 1. In Definition 4, *as a comparison and for comprehensibility*, we describe our scheme Π over parts of the encryption scheme $\Re$ shown in [41] (Section 4.2 in [41]), where there are four algorithms in $\Re$: *Setup, KeyGeneration, Encryption, and Decryption.*

Let G_1, G_2 be two multiplicative cyclic groups with prime order p. Let g be one generator of G_1 and $\dot{e}$ be a bilinear map, $\dot{e}: G_1 \times G_1 \rightarrow G_2$. We define the Lagrange coefficient $\Delta_{s,S}$ for $s \in Z_p$ and the set S of elements in Z_p [41]: $\Delta_{s,S}(c) = \prod_{t \in S, t \neq s} \frac{c-t}{s-t}$.

Definition 4 (*Attribute-based Policy Representation Scheme* Π)**.**

1. **Setup.** For the universe attributes $\mu = \{1, 2, \ldots, n\}$, the algorithm randomly chooses a number $l_s \in Z_p$ for each $s \in \mu$. The algorithm Setup in $\Re$ can be invoked to generate public parameters $pbPar$ $L_1 = g^{l_1}$, $L_2 = g^{l_2}$, …, $L_n = g^{l_n}$,$Y = \dot{e}(g, g)$; with private parameters $prPar$ l_1, …, l_n.
2. **Initialization.** For an event topic tp, the publisher i makes an access tree $T_{i,tp}$, and prepares a secret random value $\alpha_{i,tp}$ for the root node of $T_{i,tp}$, i.e., the secret $\alpha_{i,tp}$ corresponding to $T_{i,tp}$ as the secret base of policy presentation.
3. **Authorization**. Given a user j's attributes γ, the publisher i will make another access tree $T_{i,tp,j}$ for j (i.e., γ satisfying $T_{i,tp,j}$), generate a secret random value $\beta_{i,tp,j}$, and prepare $\alpha_{i,tp}\beta_{i,tp,j}$ for the root node of $T_{i,tp,j}$.

   (a) The publisher i invokes the KeyGeneration algorithm in $\Re$ with γ and $T_{i,tp,j}$ as the input, which outputs $D'_y = g^{q_y(0)/l_s}$ for each leaf node y in the tree $T_{i,tp,j}$.

   (b) The publisher i invokes the KeyGeneration algorithm in $\Re$ with γ and $T_{i,tp}$ as the input, which outputs $D_x = g^{(q_x(0)-q_x(0)\beta_{i,tp,j})/l_s}$ for $T_{i,tp}$'s leaf node x with the node attribute in γ.

   Then, $creS = \{\{D_x\}, \{D'_y\}\}$.

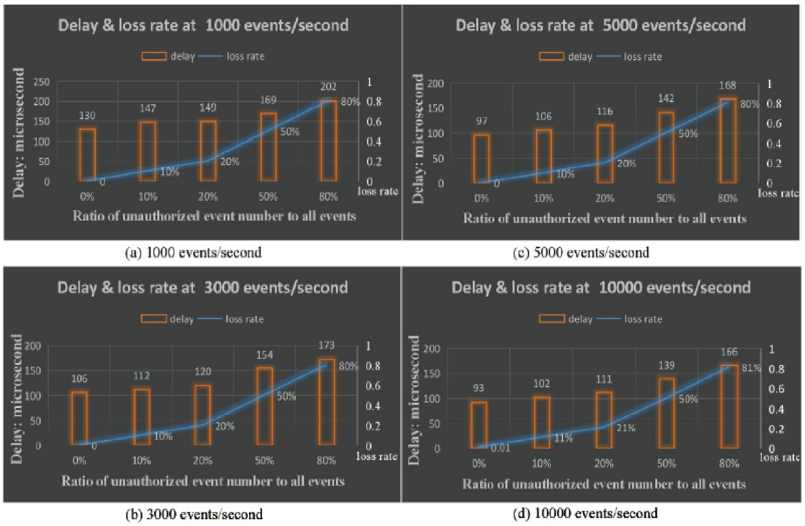


Fig. 10. Delay and loss rate at different event publishing speeds.

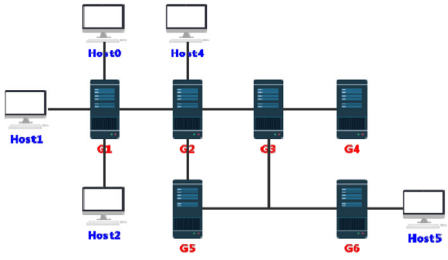


Fig. 11. Deployment environment.

4. **GeneratingRepresentation**. Given the access tree $T_{i,tp}$ and a time period, the publisher i chooses a random number r for this period, and partially invokes the Encryption algorithm $\mathfrak{R}$ only to compute $c'_k = (L_k)^r = g^{rl_k}$ for each leaf in $T_{i,tp}$ without encrypting messages. The publisher i obtains the public parameter set $par = \{c'_k\}$ of secret policy representation, which is published. The secret policy representation is $dCre = \dot{e}(g,g)^{r\alpha_{i,tp}} = Y^{r\alpha_{i,tp}}$, which is not directly published.
5. **ReconstructingRepresentation**. When the public parameter $pbPar$, representation parameter par, and authorization credentials $creS = \{\{D_x\}, \{D'_y\}\}$ are given, the secret policy representation can be computed as follows:
   (a) When the $DecryptionNode$ function in the Decryption algorithm of $\mathfrak{R}$ is invoked with $\{D_x\}$, $T_{i,tp}$, and par as input, $RT\_T_{i,tp}(root) = Y^{\alpha_{i,tp}r-\alpha_{i,tp}\beta_{i,tp,j}r}$ is output.
   (b) When the $DecryptionNode$ function in the Decryption algorithm of $\mathfrak{R}$ is invoked with $\{D'_y\}$, $T_{i,tp,j}$, and par as input, $RT\_T_{i,tp,j}(root) = Y^{\alpha_{i,tp}\beta_{i,tp,j}r}$ is output.
   (c) The secret policy representation is then computed: $RT = RT\_T_{i,tp}(root) \cdot RT\_T_{i,tp,j}(root) = Y^{r\alpha_{i,tp}}$.

The differences between our scheme Π and the encryption scheme $\mathfrak{R}$ reported in [41] are as follows:

(1) Private keys. In $\mathfrak{R}$, there is one set of private keys corresponding to the user's attributes, while, in Π, there are two sets of private keys: one corresponding to a publisher's topic issued to SDN controllers, and the other corresponding to the subscriber issued to the subscriber itself. This results in secret sharing in Π, *i.e., computing the sharing secret* $Y^{r\alpha_{i,tp}}$.

(2) Ciphertexts. In Π, ***GeneratingRepresentation*** and ***ReconstructingRepresentation*** together output a special ciphetext $c = (\gamma, \{c_i = L_i^r\}_{i\in\gamma}, \dot{Y}^{r-r\beta_{i,tp,j}}$ *or* $\dot{Y}^{r\beta_{i,tp,j}})$, where r is a random number for the current time period, $\{c_i = L_i^r\}$ is computed and published by the publisher (using ***GeneratingRepresentation***), the third element in c can be $\dot{Y}^{r-r\beta_{i,tp,j}}$ computed by SDN controllers (using ***ReconstructingRepresentation***) or $\dot{Y}^{r\beta_{i,tp,j}}$ computed by the subscriber (using ***ReconstructingRepresentation***), and $\dot{Y}^r = Y^{r\alpha_{i,tp}}$ is a sharing secret between the controller and subscriber. In $\mathfrak{R}$, the output ciphertext is $c = (\gamma, c' = m\dot{Y}^r, \{c_i = L_i^r\}_{i\in\gamma})$, while in Π no actual message m is encrypted.

5.4. Security analysis

The policy representation scheme Π is the cornerstone of our access control solution, and we prove that it is secure.

Theorem 2. *If the policy representation scheme Π is not Attribute-based Selective-Set secure, then the security of the encryption scheme $\mathfrak{R}$ is broken.*

Proof. A simulator uses the adversary of scheme Π as an inner algorithm **InA**, and simulates the adversary of $\mathfrak{R}$ to break the security of $\mathfrak{R}$. It acts as the challenger of Π to prepare for **InA** the parameters and keys of Π such as the secret random value $\beta_{i,tp,j}$ (a **security game** defined in Definition 2).

For the challenging of **InA**, the simulator submits two equal length messages m_0 and m_1 to the challenger of $\mathfrak{R}$. When the challenger of $\mathfrak{R}$ returns a ciphertext $c = (\gamma, c' = m_b\dot{Y}^r, \{c_i = L_i^r\}_{i\in\gamma})$, with $\dot{Y} = \dot{e}(g,g)^{\alpha_{i,tp}}$, the simulator uses its inner algorithm **InA** to guess. The simulator computes and guesses below.

(1) It randomly selects a bit b'(*being* 0 *or* 1)

$$c'' = (c'/(c')^{\beta_{i,tp,j}})(m_{b'})^{\beta_{i,tp,j}-1}$$

(2)
$$= (m_b\dot{Y}^r/(m_b^{\beta_{i,tp,j}}\dot{Y}^{s\beta_{i,tp,j}}))(m_{b'})^{\beta_{i,tp,j}-1}$$
$$= \dot{Y}^{r-r\beta_{i,tp,j}}(m_b^{1-\beta_{i,tp,j}}m_{b'}^{\beta_{i,tp,j}-1})$$

(3) It sends $(\gamma, \{c_i\}_{i\in\gamma}, c'')$ to **InA**.

After **InA** guesses b, the simulator may answer as follows:

(1) **A:** If $b' = b$ by **InA**, then the simulator answers b with the advantage ε by definition, i.e., having the probability $1/2+\varepsilon$.
(2) **B:** If $b' \neq b$ by **InA**, the simulator can randomly make a guess, and the probability is $1/2$.

Note, the simulator randomly guesses the bit b', and the probability $\Pr[b' = b] = 1/2, \Pr[b' \neq b] = 1/2$.

Table 2
Gain of topology construction against unauthorized nodes.

Unauthorized node ratio	10%	20%	30%	40%	50%	70%
Case a in Fig. 9	0.806	1.52	2.31	2.67	3.84	6.14
Case b in Fig. 9	1.04	1.92	3.75	3.57	3.57	8.33

The simulator breaks the security of $\Re$ with the probability:

$$\Pr[b' = b]\Pr[A|b' = b] + \Pr[b' \neq b]\Pr[B|b' \neq b] = \frac{1}{2}(\frac{1}{2} + \varepsilon) + \frac{1}{2}\frac{1}{2} = \frac{1}{2} + \frac{\varepsilon}{2}.$$

That is, when **InA** has the advantage ε to break the security of Π, the simulator has an advantage $\varepsilon/2$ to break the security of $\Re$.

There are two groups of collusion attacks on our solution. The first group of collusion attacks is that the home controller colludes with its corrupted subscribers to compute the ciphertext of the policy representations for other event topics that only non-corrupted subscribers have rights to access to. The second group of collusion attacks is that the home controller and its corrupted subscribers collect possible authorization credentials to compute the policy representations for other event topics.

Our solution resists against these collusions due to:

(1) A home controller has a part of authorization credentials. When colluding with the corrupted subscriber j, it can produce the ciphertext and the hash of the policy representation for the topic that j has rights to access to. When another subscriber j' is not corrupted, the home controller has not got all the authorization credentials for the event topic that only j' has rights to access to, and cannot reconstruct the policy representation by itself. The ciphertext and the hash of the policy representation are computed by the homomorphic functions such that the home controller cannot directly get them if it cannot break these functions.
(2) Even if it obtains multiple subscribers' authorization credentials, the controller cannot use these authorization credentials from different subscribers to obtain more authorization, because each subscriber has the authorization credentials corresponding to its own access tree, such that the mixed authorization credentials from two different subscribers make no sense for reconstructing a policy representation.

Table 2 illustrates the quantitative gain of policy-driven topology construction against unauthorized nodes, where the gain is measured by the number of unauthorized nodes that is resisted against by each extra node, and the node number of the constructed topology is normalized by treating it as the denominator in the gain. From Table 2, we know that the gain increases when the unauthorized nodes increase in the network. That is, the topology construction method is effective with one extra node against more attacks, when available network resources become less. The gain in *case b* is greater than the one in *case a* because more authorized nodes result in few extra nodes in the constructed topology, *i.e., one extra node against more unauthorized nodes with increasing the gain.*

In order to quantitatively compute the gain of the policy encoding method, we assume an authorization tree (*i.e., representing a by trees*) has n non-leaf nodes with all children having "and"/"or" relations. In general SDN-based policy enforcement methods [37], the number of flow entries is n, while, in our work, it is one, *i.e, the gain of flow space being* $n - 1$. Furthermore, the matching times averages $n/2$ in general methods, while, in our work, it is still one, the gain of computation time being $n/2 - 1$ if the policy encoding operation is periodically carried out with long time, *i.e., no considering encoding computation cost.*

6. Evaluation

This section is dedicated to the analysis of the design and implementation of the proposed access control solution for a publish/subscribe middleware over SDNs. A series of experiments are conducted to understand the effects of the design: (i) extra nodes for the publish/subscribe topology construction, (ii) event forwarding speed under the condition of making policy matching, (iii) access control delay in the whole solution, and (v) the overhead on the SDN controllers and the administrator.

6.1. Topology construction

To test the topology construction algorithm, we perform two kinds of testing, and the experiment settings are shown in Table 3. In the first kind of testing, the unauthorized node percent is fixed at 10% to unfold the authorized nodes affecting characteristics, the pub/sub node percent varies from 20% to 80%, and the connection degree of nodes ranges over [1, 3], [3, 7], and [5, 12]. In the second kind of testing, the pub/sub node percent is fixed at 20% and 40%, the unauthorized node percent varies from 10% to 70%, and the connection degree of nodes ranges over [1, 3] and [3, 7].

Fig. 8 shows the results of the first kind of testing. From Fig. 8, we know that, when the pub/sub node percent increases, the extra node number (denoted by *extraNum*; *topoNum* denoting the number of nodes to construct the topology; *pub/sub-Num* denoting the number of publishers/subscribers in the topology; and $n = 1000$) decreases. When the connection degree of the nodes increases, the number of the nodes to construct a topology for all the event streams decreases, and the extra node number decreases. Fig. 9 shows the results of the second kind of testing. In Fig. 9, it is interesting that the unauthorized node percent almost does not affect the topology construction. When the unauthorized node percent increases, the extra node number almost remains unchanged. These experimental results show that our topology construction is effective.

6.2. Policy matching

In the experiments of testing the event forwarding speed under the condition of policy matching, we use physical SDN switches (Pica8 P-3290: 512 MB Memory, MPC8541 CPU, 1 μs/64 Byte-Frames, 48 Ports with 10/100/1000BASE-T), where a publisher publishes events at different speeds measured by events per second: 1000 events/second, 3000 events/second, 5000 events/second, and 10 000 events/second, with each event including 1024 bytes.

Fig. 10 illustrates the event forwarding speed of SDN switches under the condition of matching policy representations, where the number of events without the correct authorization policy representation varies at different ratios of 0, 0.1, 0.2, 0.5, and 0.8. From Fig. 10, we know that the event loss rate almost is the same as the ratios of the incorrect input events after the matching operations of policy representation are imposed on event streams. The extra delay from policy matching is about 0.88%. That is to say, using policy representation to encode policies into flow tables is an effective way to establish the access control in the publish/subscribe middleware over SDNs.

6.3. Performance comparison

In order to compare ours with others, two kinds of experiments have been conducted, where the experiment environment in Fig. 10 is used, the SDN network deployment environment is illustrated in Fig. 11, and we set 10 rules (ten attribute conjunctions/disjunctions) and 20 attributes (using them to randomly form rules and determining authorization nodes) respectively as the experimental parameters. In the first kind of experiments, we test the performance of a general solution without access control capabilities, which is used as the comparison basis. In the second kind of experiments, the delay for different event publishing rates is tested for ours and the access control solution of DCACF in [50].

In Fig. 12, the event publishing speed is set from 200 events/second to 5000 events/second for testing the whole solution with different

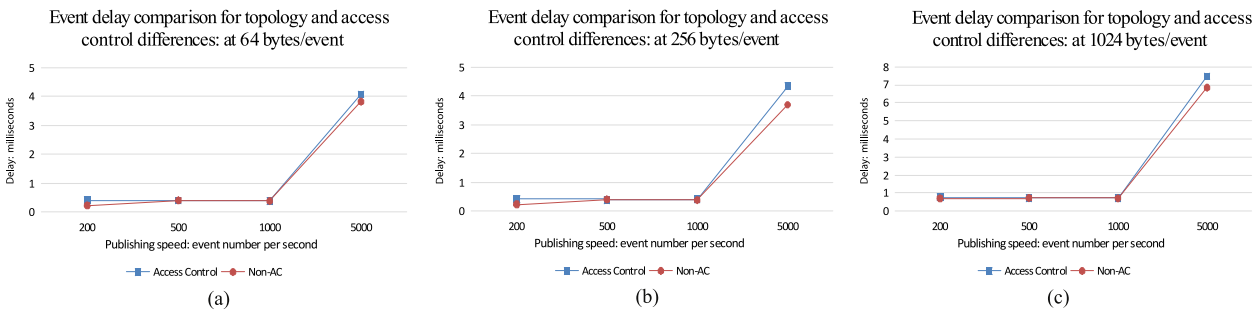


Fig. 12. Event delivery delay for topology and access control differences: Non-AC meaning no access control in middlewares.

Table 3
Settings of topology construction.

n	The number of network nodes: [100, 1000]	k	The number of event streams
m	The number of publishers & subscribers	*pub/sub percent*	m/n = 20%, 30%, 40%, 50%, 70%, 80%
u	The number of unauthorized nodes	*unauthorized percent*	u/n = 10%, 20%, 30%, 40%, 50%, 70%
deg	The connection degree of nodes: [1, 3], [3, 7], [5, 12]	m/k	The number of nodes in each event stream
$m/k * 20\%$	The number of publishers in each event stream	$m/k * 80\%$	The number of subscribers in each event stream
cost	Link cost: (0, 100]	$n * cost$	The cost on links out of streams and unauthorized nodes

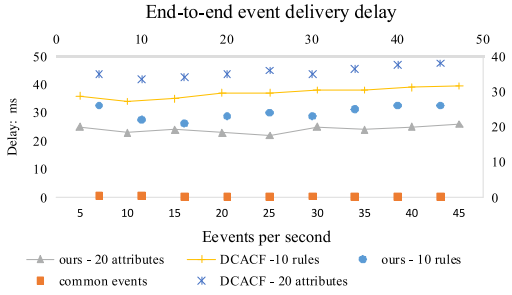


Fig. 13. End-to-end delay with encryption.

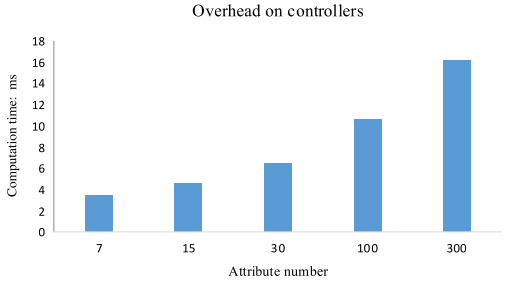


Fig. 14. Overhead on SDN controllers.

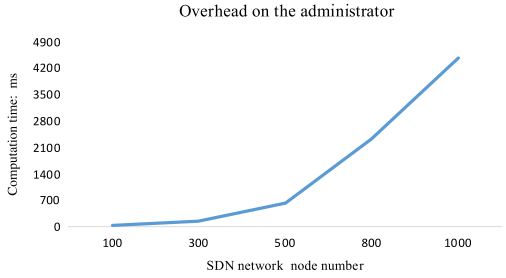


Fig. 15. Overhead on the administrator.

event sizes. Fig. 12 shows that the end-to-end event delivery delay increases when the policy matching is carried out in the home SDN switches, and the constructed publish/subscribe topology is used. But the performance of our solution is comparable to the general one without access control operations. When the event size and publishing speed increase, the delay increases with rapid ascension after congestion. The delay is greater than the one in Fig. 10 because the former is tested as a whole, and the later is only tested by the forwarding cost on one SDN switch without other costs being considered.

In Fig. 13, the confidentiality mechanism is integrated, and the access control solution of DCACF [50] based on encryption schemes is compared with ours. Fig. 13 shows that the end-to-end event delivery delay in ours is smaller than the one of DCACF [50], although the same encryption scheme is used by the two solutions. It is due to that, in our solution, policies are encoded into flow entries before the event delivery, and only one encryption operation is carried out during the event delivery. In addition, the difference of the policies almost makes no effects on the delay because of policy encoding in advance. Although our method satisfies the real-time requirements of most IoT applications, to accomplish the line-rate event delivery, special encryption devices are needed when confidentiality is integrated.

6.4. Overhead evaluation

Fig. 14 illustrates the overhead of computing the policy representation on SDN controllers, where the overhead is measured by the time spent for the policy representation computation, and the number of attributes is set from 7 to 300 with randomly forming rules. Fig. 15 illustrates the overhead of computing the global publish/subscribe topology for many topics on the administrator, where the overhead is measured by the time spent for the topology construction, and the topology construction parameters are set as in Table 3. The two figures show that our solution is efficient with the computation time staying at the level of near sub-seconds, when the publish/subscribe middleware has a moderate scale. In addition, the computation on the controllers and the administrator does not happen frequently, and concurrency optimization is possible when the scale becomes larger.

7. Conclusions

How to use SDN switches and SDN controllers to implement the access control of publish/subscribe middlewares over SDNs is the main issue addressed in this paper. Existing work on publish/subscribe middlewares over SDNs did not comprehensively explore the issue, and the access control solutions over traditional networks did not shed light on the matching capabilities of flow tables of SDN switches. We

firstly propose a method to construct a connected publish/subscribe topology based on authorization policies, which enhances our policy enforcement at the global network layer rather than on home switches to check policies. The matching capabilities of SDN switches are then used to compare the authorization policies with the subscribers' attributes, where complex authorization policies are flattened and encoded into the bit string with high data compression and security. A two-party protocol is designed to resist against the collusion attacks of SDN controllers and subscribers. These three schemes form our cross-layer access control solution for a publish/subscribe middleware over SDNs, which avoids over-delegation and over-user-controlling, and maintains a flexibility and a balance of performance and security. Finally, the evaluation shows that our design is effective.

In the future, we will further improve the scalability of our solution although we have evaluated its efficiency at the scale of one thousand of nodes in a network. There are two directions for this issue. The first direction is to use a more efficient Steiner tree generation algorithm as the basis on which the topology construction method is imposed by authorization policies. The second direction is to exploit the potential of topic name tree to further reduce the number of flow entries and matching times, where one topic name will not be independent with others in the same topic tree, and a parent topic can cover its children topics such that one policy representation of the parent may represent multiple others in some scenarios (e.g. treating a cluster of nodes as one event broker). Others such as streamlining policy matching are also available.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (no. 61372115), the National Key Research and Development Program of China (No. 2018YFB1003800), and EU H2020 DOMINOES Project (No. 771066).

References

[1] A. Hakiri, P. Berthou, A. Gokhale, S. Abdellatif, Publish/subscribe-enabled software defined networking for efficient and scalable IoT communications, IEEE Commun. Mag. 53 (9) (2015) 48–54.
[2] A. Carzaniga, D.S. Rosenblum, A.L. Wolf, Design and evaluation of a wide-area event notification service, ACM Trans. Comput. Syst. 19 (3) (2001) 332–383.
[3] H.-A. Jacobsen, A.K.Y. Cheung, G. Li, B. Maniymaran, V. Muthusamy, R.S. Kazemzadeh, The PADRES publish/subscribe system, in: Principles and Applications of Distributed Event-Based Systems, 2010, pp. 164–205.
[4] M.A. Tariq, B. Koldehofe, G.G. Koch, I. Khan, K. Rothermel, Meeting subscriber-defined QoS constraints in publish/subscribe systems, Concurr. Comput.: Pract. Exper. 23 (11) (2011) 2140–2153.
[5] O.M.E. Committee, Software-defined Networking: The New Norm for Networks. Open Networking Foundation, 2012.
[6] D.E. Bakken, A. Bose, C.H. Hauser, D.E. Whitehead, G.C. Zweigle, Smart generation and transmission with coherent, real-time data, Proc. IEEE 99 (6) (2011) 928–951.
[7] P. Porras, S. Shin, V. Yegneswaran, M. Fong, M. Tyson, G. Gu, A security enforcement kernel for OpenFlow networks, in: Proc. of the 1st Workshop on Hot topics in Software Defined Networks, ACM, Helsinki, 2012, pp. 121–126.
[8] P. Porras, S. Cheung, M. Fong, K. Skinner, V. Yegneswaran, Securing the software-defined network control layer, in: Proc. of the 2015 Annual Network and Distributed System Security Symp. (NDSS 2015), Internet Society, San Diego, 2015, pp. 1–15.
[9] H. Wang, L. Xu, G. Gu, FloodGuard: A DoS attack prevention extension in software-defined networks, in: Proc. of the 45th Annual IEEE/IFIP, Int'l Conf. on Dependable Systems and Networks (DSN 2015), Rio de Janeiro, 2015.
[10] I. Alsmadi, D. Xu, Security of software defined networks: A survey, Comput. Secur. 53 (2015) 79–108.
[11] J. Bacon, D.M. Eyers, J. Singh, P.R. Pietzuch, Access control in publish/subscribe systems, in: Proceedings of the Second International Conference on Distributed Event-based Systems, ACM, 2008, pp. 23–34.
[12] OMG. 2014. http://www.omg.org/spec/DDS-SECURITY/1.0/Beta1/PDF/, 2014.
[13] M. Ion, G. Russello, B. Crispo, Design and implementation of a confidentiality and access control solution for publish/subscribe systems, Comput. Netw. 56 (7) (2012) 2014–2037.
[14] S. Müller, S. Katzenbeisser, Hiding the policy in cryptographic access control, in: International Conference on Security & Trust Management, 2011, pp. 90–105.
[15] M.A. Tariq, B. Koldehofe, K. Rothermel, Securing broker-less publish/subscribe systems using identity-based encryption, IEEE Trans. Parallel Distrib. Syst. 25 (2) (2014) 518–528.
[16] G. Chockler, R. Melamed, Y. Tock, R. Vitenberg, Constructing scalable overlays for pub-sub with many topics: problems, algorithms, and evaluation, in: Twenty-sixth ACM Symposium on Principles of Distributed Computing, 2007, pp. 109–118.
[17] Ch. Chen, H.-A. Jacobsen, R. Vitenberg, Algorithms based on divide and conquer for topic-based publish/subscribe overlay design, IEEE/ACM Trans. Netw. 24 (1) (2016) 422–436.
[18] M. Onusa, A.W. Richab, Parameterized maximum and average degree approximation in topic-based publish–subscribe overlay network design, Comput. Netw. 94 (2016) 307–317.
[19] M. Castro, P. Druschel, A.M. Kermarrec, A.I. Rowstron, SCRIBE: A large-scale and decentralized application-level multicast infrastructure, IEEE J. Sel. Areas Commun. 20 (8) (2002) 1489–1499.
[20] S.Q. Zhuang, B.Y. Zhao, A.D. Joseph, R.H. Katz, J.D. Kubiatowicz, Bayeux: An architecture for scalable and fault-tolerant wide-area data dissemination, in: Proceedings of the 11th International Workshop on Network and Operating Systems Support for Digital Audio and Video, Port Jefferson, NY, USA, 2001, pp. 11–20.
[21] R. Baldoni, R. Beraldi, V. Quema, L. Querzoni, S. Tucci-Piergiovanni, TERA: topic-based event routing for peer-to-peer architectures, in: Proceedings of the 2007 Inaugural International Conference on Distributed Event-Based Systems, Toronto, ON, Canada, 2007, pp. 2–13.
[22] V. Ramasubramanian, R. Peterson, E.G. Sirer, Corona: a high performance publish–subscribe system for the World Wide Web, in: Proceedings of the Symposium on Networked Systems Design and Implementation, Vol. 6, San Jose, CA, USA, 2006, pp. 115–117.
[23] S. Banerjee, B. Bhattacharjee, C. Kommareddy, Scalable application layer multicast, in: SIGCOMM'02 Proceedings of the 2002 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, 2002, pp. 205–217.
[24] Muhammad Adnan Tariq, Boris Koldehofe, Sukanya Bhowmik, Kurt Rothermel, PLEROMA: A SDN-based high performance publish/subscribe middleware, in: ACM/IFIP/USENIX Middleware Conference, 2014, pp. 217–228.
[25] Akram Hakiri, Aniruddha Gokhale, Data-centric publish/subscribe routing middleware for realizing proactive overlay software-defined networking, in: DEBS'16 Proceedings of the 10th ACM International Conference on Distributed and Event-based Systems, 2016, pp. 246–257.
[26] S. Bhowmik, M.A. Tariq, L. Hegazy, K. Rothermel, Hybrid content-based routing using network and application layer filtering, in: IEEE 36th International Conference on Distributed Computing Systems (ICDCS), 2016, pp. 221–231.
[27] S. Bhowmik, M.A. Tariq, J. Grunert, K. Rothermel, Bandwidth-efficient content-based routing on software-defined networks, in: DEBS' 16 Proceedings of the 10th ACM International Conference on Distributed and Event-based Systems, 2016, pp. 137–144.
[28] Y.L. Wang, Y. Zhang, J.L. Chen, SDNPS: A load-balanced topic-based publish/subscribe system in software-defined networking, Appl. Sci. 6 (4) (2016) 1–21, 91.
[29] L. Opyrchal, A. Prakash, A. Agrawal, Supporting privacy policies in a publish–subscribe substrate for pervasive environments, J. Networks 2 (1) (2007) 17–26.
[30] P. Pal, G. Lauer, J. Khoury, N. Hoff. J. Loyall, P3S: A privacy preserving publish subscribe middleware, in: Middleware 2012, Springer, 2012, pp. 476–495.
[31] S. Choi, G. Ghinita, E. Bertino, A privacy-enhancing content-based publish/subscribe system using scalar product preserving transformations, in: Database and Expert Systems Applications, Springer, 2010, pp. 368–384.
[32] W.X. Rao, L. Chen, S. Tarkoma, Toward efficient filter privacy-aware content-based pub/sub systems, IEEE Trans. Knowl. Data Eng. 25 (11) (2013) 2644–2657.
[33] J. Bethencourt, A. Sahai, B. Waters, Ciphertext-policy attribute-based encryption, in: Security and Privacy, 2007. SP'07. IEEE Symposium on, IEEE, 2007, pp. 321–334.
[34] H. Padekary, Y. Parky, H. Huz, S.-Y. Changx, Enabling dynamic access control for controller applications in software-defined networks, in: ACM on Symposium on Access Control MODELS and Technologies, 2016, pp. 51–61.
[35] F. Klaedtke, G.O. Karame, R. Bifulco, H. Cui, Access control for SDN controllers, in: ACM Sigcomm Workshop on Hot Topics in Software Defined Networking, 2014, pp. 219–220.
[36] S. He, J. Liu, J. Mao. J. Chen, Hierarchical solution for access control and authentication in software defined networks, Int. Conf. Netw. Syst. Secur. 8792 (2014) 70–81.
[37] J.Q. Liu, Y. Li, H.D. Wang, D.P. Jin, L. Su, L.G. Zeng, T. Vasilakos, Leveraging software-defined networking for security policy enforcement, Inform. Sci. 327 (2016) 288–299.
[38] S. Pisharody, J. Natarajan, A. Chowdhary, A. Alshalan, D. Huang, Brew: A security policy analysis framework for distributed SDN-based cloud environments, IEEE Trans. Dependable Secure Comput. PP (99) (2017) 1-1.
[39] Open Networking Foundation. OpenFlow management and configuration protocol (OF-CONFIG v1.1.1). Technical report, Mar. 2013.
[40] Open vSwitch. http://openvswitch.org/.
[41] V. Goyal, O. Pandey, A. Sahaiz, B. Waters, Attribute-based encryption for fine-grained access control of encrypted data, in: Proceedings of the 13th ACM Conference on Computer & Communications Security, 2006, pp. 89–98.

[42] Y.-R. Chen, S. Radhakrishnan, S. Dhall, S. Karabuk, On multi-stream multi-source multicast routing, Comput. Netw. 57 (2013) 2916–2930.
[43] S.L. Hakimi, Steiner's problem in graphs and its implications, Networks 1 (1971) 113–133.
[44] R.M. Karp, Reducibility among combinatorial problems, in: R.E. Miller, J.W. Thatcher (Eds.), Complexity of Computer Computations, Plenum, New York, NY, 1972, pp. 85–103.
[45] E.W. Dijkstra, A note on two problems in connexion with graphs, Numer. Math. 1 (1959) 269–271.
[46] L. Kou, G. Markowsky, L. Berman, A fast algorithm for Steiner trees, Acta Inform. 15 (2) (1981) 141–145.
[47] G.L. Nemhauser, L.A. Wolsey, Integer and Combinatorial Optimization, Wiley, ISBN: 978-0-471-82819-8, 1998.
[48] C. Gentry, Fully homomorphic encryption without bootstrapping. http://eprint.iacr.org, 2011.
[49] Y. Zhang, J.L. Chen, Wide-area SCADA system with distributed security framework, J. Commun. Netw. 14 (6) (2012) 597–605.
[50] L. Duan, D.X. Liu, Y. Zhang, Sh. P. Chen, R.P. Liu, B. Cheng, J.L. Chen, Secure data-centric access control for smart grid services based on publish/subscribe systems, ACM Trans. Internet Technol. 0 (0) (2016) 23, pp. 1–17, Vol. 16, Issue 4.
[51] Y.S. Kim. J. Heo, Device authentication protocol for smart grid systems using homomorphic hash, J. Commun. Netw. 14 (6) (2012) 606–613.
[52] E.L. Gazzoni, D. Luiz, G. Filho, E. Politécnica, Demonstrating data possession and uncheatable data transfer. IACR Cryptology ePrint Archive, 2006.
[53] M.N. Krohn, M.J. Freedman, D. Mazieres, On-the-fly verification of rateless erasure codes for efficient content distribution, in: IEEE Symposium on Security and Privacy, 2004, pp. 226–240.
[54] F.E. Oggier, A. Datta, Self-repairing homomorphic codes for distributed storage systems, in: 30th IEEE International Conference on Computer Communications, 2010, pp. 1215–1223.