2016 IEEE 24th International Conference on Network Protocols (ICNP)
Workshop on COntrol, Operation and appLication in SDN protocols (CoolSDN 2016)

SDIG: Toward Software-Defined IPsec Gateway

Wei Li, Fengxu Lin and Guanchao Sun
Key Lab of Beijing Network Technology, School of Computer Science and Engineering
Beihang University, Beijing, China
Email: {liw, linfx7, gcsun}@buaa.edu.cn

***Abstract*—The current IPsec gateway integrates many functions of IPsec operation, tunnel management and forwarding decision, which makes the IPsec gateway complicated in maintenance and deployment. The problem of maintaining such devices prevents IPsec VPN from applying widely. The emergence of SDN provides an innovative way to decouple the control plane and data plane. In this paper, a Software-Defined IPsec Gateway (SDIG) is proposed to achieve net2net IPsec VPN. Different from the traditional IPsec gateway, the SDIG device serves as a data plane equipment that just concentrates on exchanging IKE packets and encrypting/decrypting IP packets. A global view of SDIG devices can be constructed in the SDN controller by collecting the status of all devices. Therefore the controller can manage and configure SDIG devices centrally, and simplify deployment complexity. Outbound IP packets for the SDIG device can be viewed as a trigger to control the establishment of IPsec tunnels. The SDIG device and the controller exchange information through a customized southbound protocol. The prototype system of SDIG is implemented, and the preliminary experimental results show that the method is feasible and effective.**

I. Introduction

Recent years, Software-Defined Networking (SDN) has received a lot of attention in enterprise network. SDN architectures decouple network control and forwarding functions, enabling network intelligence centralized in SDN controllers which maintain a global view of the network. With SDN, network can be controlled from a single point, which simplifies the network management. The network devices are also simplified, since they no longer need to make forwarding decisions but accept instructions from controllers.

Internet Protocol Security (IPsec) is a protocol suite designed to protect IP communications against cyber threats. IPsec-based VPN solutions work on IP layer and provide application-independent service. These features make IPsec one of the most popular choices in the industry. Meanwhile, the complexity of IPsec increases, which makes the IPsec gateway difficult to manage. The complexity of IPsec VPN is highly concentrated in the IPsec gateway. This paper is motivated by the shortcomings of the current gateway.

- **Integrate too many functions.** The current IPsec device integrates many functions: IKE exchange, AH/ESP process, tunnel management and forwarding decision. Administrators must learn professional knowledge to configure the gateway correctly.
- **Rigid tunnel management.** The establishment of IPsec tunnel is triggered by administrators' instruction. The IPsec Gateway is unable to protect traffic without preconfiguring.
- **Static SPD management.** The current IPsec device performs a static policy management. Any unknown traffic will be discarded. The gateway just waits for administrators to "push" configurations.

In this paper, a Software-Defined IPsec Gateway (SDIG) is proposed to simplify the IPsec management and perform flexible control. The proposed SDIG is mainly used to achieve net2net IPsec VPN.

Different from the traditional IPsec gateway, the SDIG device serves as a data plane equipment that just concentrates on exchanging IKE packets and encrypting/decrypting IP packets. A global view of SDIG devices can be constructed in the SDN controller by collecting the status of all devices. By using IPsec connections and forwarding rules, the controller takes over tunnel management and forwarding decision functions from gateways. Therefore the controller can manage and configure SDIG devices centrally. Outbound IP packets for the SDIG can be viewed as a trigger to control the establishment of IPsec tunnels. The SDIG device and the controller exchange information through a customized southbound protocol.

Specifically, we make the following contributions: 1) The SDIG architecture is proposed to decouple the function of IPsec gateway. 2) A customized southbound protocol is designed to exchange information between the SDIG device and the SDN controller. 3) We implement a prototype of SDIG and analyze the feasibility and effectiveness through experiments.

The rest of this paper is organized as follows. The motivation of our work is described in Session 2. In Session 3 we describe the system design of SDIG. We implement a prototype of the system and make preliminary experiments to test the system, these works are described in Session 4. Related work is introduced in Session 5. Finally, we conclude the paper in Session 6.

II. Problem Statement

A. Traditional IPsec Gateway Model

IPsec gateways are deployed at the edge of network, and usually have two interfaces that connect to the Internet and the enterprise network separately. The one connected to the Internet is an unprotected interface and packets delivered through the interface should be encrypted.

An IPsec gateway must implement AH and ESP in order to serve a set of internal hosts, providing security services for these hosts when they communicate with external hosts also employing IPsec.

978-1-5090-3281-5/16/$31.00 ©2016 IEEE

The shortcomings of such gateways mainly centralized in the way tunnels are build and the method SPD is managed.

1) Building Tunnels in Old Way: IPsec tunnels consist of Security Associations (SAs) [1]. An SA is a simplex connection between two gateways. Security services are afforded to an SA by the use of AH or ESP. While using SAs to enforce security policy for traffic crossing the IPsec boundary, Internet Key Exchange (IKE) [2] is used to establish and maintain SAs. IKE performs mutual authentication between two peers and establishes an IKE SA. The IKE SA consists of shared information that can be used to establish Child SAs for AH and ESP during further IKE exchange, as well as a set of cryptographic algorithms used for IPsec processing.

During IKE exchange, traffic selectors (TSs) are negotiated. TSs specify the selection criteria for packets which will be forwarded over the newly set up SA. TSi and TSr, which respectively correspond to addresses of subnets behind initiator and responder, are proposed by initiator in request message. Responder decides TSs depending on its configuration values and then sends the result back.

The establishment of IKE SA (IPsec tunnel) is triggered by administrators' instruction in the traditional gateway. TSs are determined using the pre-configured parameters. This makes the gateway a rigid device which is unable to protect traffic without pre-configuring. A new method to establish IPsec tunnels is needed to perform more flexible tunnel management.

2) Managing Security Policies with Legacy Method: Security policies specify what processes will be performed to packets and in what fashion. In the IPsec gateway, they are stored in the Security Policy Database (SPD) [1] which must be consulted during the processing of inbound and outbound traffic. The SPD consists of ordered entries, just like packet filters in routers. For any inbound or outbound traffic, there are three choices available: discard, bypass IPsec or protect IPsec. Discard means that the packet is not allowed to traverse the IPsec boundary. The second choice refers to traffic that is allow to cross the IPsec boundary but without IPsec protection. The last one means apply IPsec processing. For traffic that is to be protected with IPsec, the SPD entry consists of selectors that apply to the IPsec-protected traffic, controls on how to create SAs based on these selectors, and the parameters needed to effect this protection.

For traditional gateways, the default operation for traffic is discard, if there are no SPD entries found. This leads the gateway to a static policy management. The gateway just waits for administrators to "push" configurations. We improve the gateway using a customized default operation for traffic: report to controller, which enables the gateway to "pull" policies actively.

B. SDN Brings Innovations

SDN white paper [3] from Open Network Foundation (ONF) describes SDN with a three-tier architecture. Network intelligence is centralized in software-based SDN controllers which maintain a global view of the network. As a result, network devices appear to be simplified switches, and no longer need to understand and process thousands of protocol standards but merely accept instructions from SDN controllers. Once needed, network administrators can configure the centralized network abstraction other than operating tens of thousands of lines of configuration scattered among thousands of devices.

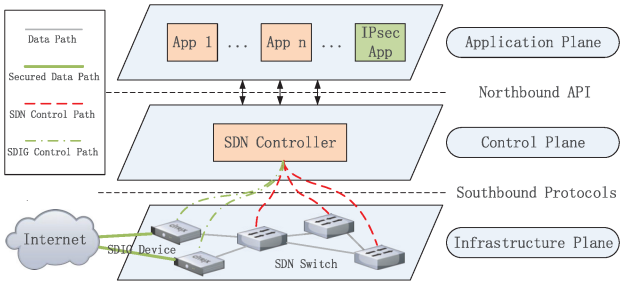


Fig. 1. SDIG Devices in SDN Architecture

The interface controllers use to communicate with network devices is known as southbound interface. OpenFlow [4], originally designed to provide a way for researchers to run experimental protocols in the networks they use, receives wide attention and becomes the major standard of southbound protocols. OpenFlow enables controllers to manipulate network data paths using flow table, configuring and managing forwarding devices are performed with OF-CONFIG protocol.

Many controllers, such as Beacon [5], Floodlight [6], POX [7], Ryu [8] and ONOS [9], take OpenFlow as the sole southbound protocol. The OpenDaylight SDN controller [10] extends the basic premise of SDN and enables support of multiple southbound protocols via plugins.

The emergence of SDN provides us a innovative way to decouple control panel and data panel of the IPsec gateway. And the OpenDaylight enables us to custom southbound protocol and perform centralized management using the SDN controller.

III. System Design

The SDIG consists of an SDIG device which runs as an IPsec gateway and an SDN controller which manages SDIG devices centrally. As is shown in Figure 1, SDIG devices lie on the infrastructure plane same as SDN switches. The controller has a global view of the network, and manages all the network devices centrally. The controller provides interfaces for administrators to programmatically configure SDN switches as well as SDIG devices.

The controller communicates with network devices through control paths. While SDN control path uses existing protocol such as OpenFlow, SDIG control path uses our customized southbound protocol.

SDN switches forward general packets, and SDIG devices handle packets that will be encrypted or decrypted. When packets arrive, the SDIG device will look up its SPD to decide

what to do with the packets, like the SDN switch and the flow table.

With the traditional method, decisions are made from fixed configuration parameters. Administrators have to pre-configure security gateways to satisfy the security requirement. But for SDIG devices, all the gateways are managed by controllers, all the forwarding decisions are made by controllers. SDIG devices just concentrate on exchanging IKE packets and encrypting/decrypting IP packets. In this way, the intelligence of SDIG devices is decoupled and concentrated in the SDN controller.

We decouple IPsec gateway functions by separating them into two parts: management functions that consists of maintaining IPsec tunnels and making forwarding decision, basic functions that perform fundamental IPsec processes. In our design, management functions are taken over by SDN controllers.

Managing IPsec tunnels is managing configuration parameters used to build tunnels and determining the establishment of tunnels. Forward decision function is the ability to decide what to do with incoming packets. Between all the configuration parameters used to build tunnels, leftsubnet and rightsubnet are special ones. Each IPsec implementation needs users to pre-configure the leftsubnet which indicates the subnet address behind the implementation, and the rightsubnet which indicates the other end of the subnet. During IKE exchange, leftsubnet and rightsubnet are used to generate TSi and TSr. The generated TSi and TSr are negotiated and determined through TS negotiation. Forward decisions are made mainly based on TSs. Configuration parameters except for leftsubnet and rightsubnet are used to build IPsec connections. The introducing of IPsec connections enables SDN controllers to perform centralized management of configuration parameters. IPsec rules consisting of leftsubnet, rightsubnet and an action field are used by controllers to make forwarding decision. IPsec rules and IPsec connections work together to decide whether to build IPsec tunnels.

A. SDN Controller for SDIG

In this session, we describe how the SDN controller manage SDIG devices centrally.

As the brain of SDIG devices, the controller must collect status information regularly. Using the collected information, the controller builds a global view of all the SDIG devices. Since the controller knows the resource consumption on every SDIG device, it can make decisions depending on the overall situation.

To manage SDIG devices is to perform tunnel management and make forwarding decisions. Two elements are maintained by the SDN controller to act management functions: IPsec connections that are used for setting up tunnels (tunnel management), and forwarding rules that specify what to do to packets (forwarding decision). These elements are configured and saved in the controller, and will be issued to SDIG devices when they are needed.

1) Managing Tunnels with IPsec Connections: An IPsec connection is a set of configuration parameters used for setting up IPsec tunnels. There can be multiple connections in a controller, with each connection having a unique ID. Every connections consists of a series of configuration parameters. These parameters provide value for but not limited to following items: IKE version, algorithms for AH/ESP, authentication method, ip addresses of gateways.

There are two kinds of connections which correspond to active mode and passive mode of the SDIG. Active connections are used to establish IPsec tunnels actively, while passive connections are used to wait for other gateways to establish tunnels.

A list of configuration parameters for traditional IPsec connections (connections in strongSwan, an opensource IPsec VPN implementation) can be found in [11]. Most parameters of our IPsec connections are the same as ones in strongSwan, except for leftsubnet and rightsubnet. During IKE exchange, leftsubnet and rightsubnet are used to negotiate TSs which specify the selection criteria for packets which will be forwarded over the newly set up tunnel. In active connections, their function is replaced by forwarding rules and they will not show up in configuration parameters. In passive connections, leftsubnet and rightsubnet are set to "0.0.0.0/0", which means leaving them for initiators of IKE exchange to decide.

Both of the connections are configured in controllers, and issued to gateways to take effect. After being configured, passive connections will be issued to security gateways at once. For active connections which cannot be used alone, they must be used together with forwarding rules. The selector field of forwarding rules will be used as a complement of leftsubnet and rightsubnet. In this way, IPsec tunnels are managed by the controller.

2) Maintaining Security Policies with Forwarding Rules: IPsec connections define what IPsec tunnels should be like, while forwarding rules indicate which active connection will be used to build tunnels for an outbound packet to go through. Inside controllers, all forwarding rules are saved in a ordered table which is consistent with packet filters in firewalls.

A forwarding rule consists of a selector field and an action field. The selector field has two subnet addresses, one for address packets coming from and another for address packets going to. There are three kinds of actions in our rules: apply IPsec, bypass IPsec, and discard. When using apply IPsec, connection ID must be specified to indicate which active connection to use.

Controllers can take the initiative to issue forwarding rules to gateways, or issue them after receiving packet notifications and finishing forwarding rules lookup. The issued rules are used by SDIG devices to generate their SPD entries, and the issued IPsec connections are used to establish tunnels. This enables the controller to take over forwarding decision functions.

3) Reducing Manual Configuration: The introduction of IPsec connection and IPsec rule not only provides the way

to perform centralized management but also gives a method to reduce manual configuration of IPsec VPN.

Reusing IPsec connections is the key to reduce manual configuration. In old ways, you must configure every necessary parameter for an IPsec tunnel, even though most of them are the same as previous tunnels. In our design, the most likely modified parameters in IPsec configuration are extracted and used to build IPsec rules, with the rest parameters becoming IPsec connections. In many cases, administrators just need to add an IPsec rule to build a new tunnel, instead of configuring whole parameters.

Here gives an example. IPsec tunnels are built between Intranet A (10.1.0.0/16) and Intranet B (10.2.0.0/16). An existing tunnel has been built with IPsec rule R1 and IPsec connection C1. R1 specifies that packets from 10.1.1.0/24 to 10.2.1.0/24 should be protected with C1. And C1 specifies gateway addresses, encryption algorithms and other parameters that are needed to build tunnels. The existing tunnel allows 10.1.1.0/24 to communicate with 10.2.1.0/24. Now, another tunnel between 10.1.2.0/24 and 10.2.2.0/24 is needed. What the administrator needs to do is adding an IPsec rule which indicates that packets from 10.1.2.0/24 to 10.2.2.0/24 should be protected with C1. In this way, IPsec connections are reused.

B. How SDIG Device Works

An SDIG device is a special IPsec gateway which only provides IKE exchange and IP packets encryption/decryption services. The SDIG device must be used together with the SDN controller. The SDIG device does not manage IPsec tunnels itself or make forwarding decisions. Instead, an SDN controller is used to perform centrally management. All the functions are performed with the controller, administrators do not need to access security gateways one by one to make such devices work.

The SDIG device only runs as net2net VPN gateway to achieve interconnection between subnets, and works in both active mode and passive mode.

1) Processing Incoming Packets: When a packet arrives, the SDIG device will perform an SPD lookup. The SPD used here is derived from forwarding rules, and the contains three possible choices: apply IPsec, bypass IPsec, discard.

The gateway takes report to controller as default action for packets, in case there are no SPD entries found for them. When a gateway fails in deciding what to do with a packet, a digest of the packet will be sent to the controller. The controller will make a decision and issue the result (rule, with IPsec connections if necessary).

There are four kinds of packets the SDIG device processes:

1) Packet from unprotected network to the gateway. If the packet is an IKE packet, it will be processed with IKE procedure. If the packet is an IPsec protected packet, it will trigger an SAD lookup and be proceed with found SAs. The packet will be discarded if there are no SAs found.

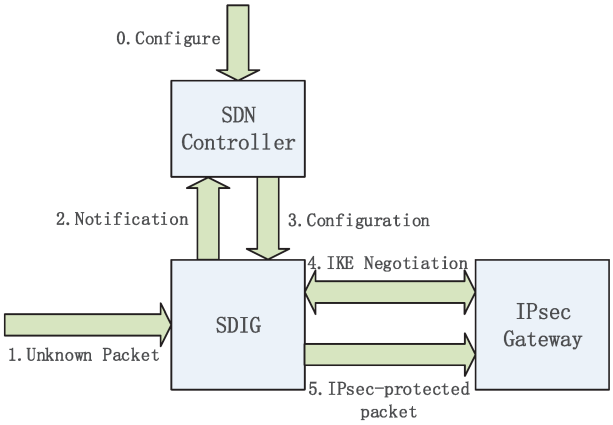


Fig. 2. Process with Unknown Packets

2) Packet from unprotected network to the protected network behind the gateway. When such packet arrives, an SPD lookup will be performed and decide whether to bypass IPsec and forward to the protected network.
3) Packet from protected network to the gateway. Such packet is usually used for gateway management and will be dealt with by the gateway itself. If the packet is not for management, it will be discarded.
4) Packet from protected network to unprotected network. A packet from protected interface will trigger an SPD lookup. If there are suitable SPD entries for the packet, it will be processed as the entries specify. If not, it will trigger the process described in Session III-B2.

2) Working with SDN Controller: SDIG devices are subject to the SDN controller when they are working, and need to send status information to controller regularly. When an SDIG device overloads, crashes, or receives outbound packets that cannot be dealt with, it will notice the controller such events. SDIG devices also receive configuration and query messages from controllers.

Passive IPsec connections are pre-issued by controller to SDIG, so that the SDIG device can response IKE negotiations without delay.

For active IPsec connections, the way IPsec tunnels are built is quite different from traditional ones. The establishment of IPsec tunnels is trigger by packet from protected network to unprotected network. Usually, such packet is processed as the SPD specified. But if there are no SPD entries found for the packet, process shown in Figure 2 will be triggered. A digest of packet, including source ip and destination ip, will be sent to the controller. The controller will lookup its forwarding rules to get a suitable rule for the packet, and issue the found rule. If needed, IPsec connections used by the rule will be issued as well.

C. Customizing Southbound Protocol

In this session, we describe the customized southbound protocol that is used for exchanging information between SDN controllers and SDIG devices.

TABLE I
MESSAGE PAYLOAD NOTATIONS

Notation	Payload
HDR	Message header
IDc	Identification - Controller
IDg	Identification - Gateway
NT	Notification type
ET	Element type
DES	Description
RTN	Return

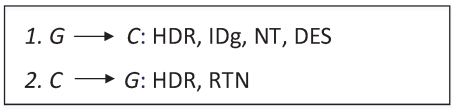


Fig. 3. The Notification

The protocol includes three kinds of communications: notifications, configuration messages and query messages. Each of the communications consists of a pair of messages: a request and a response. All the messages are transported through TCP or TLS.

In the following descriptions, payloads contained in the message are indicated by names as listed in Table I, and payloads that may be repeated zero or more times will be shown in curly braces, such as {ET}.

1) The Notification: Notifications are pairs of messages: a notification message and a response message. All notifications are initiated by the gateway (SDIG device) and responded by the controller as is shown in Figure 3. The gateway uses notification messages to send status and events to the controller. The controller receiving notification messages must reply with response messages.

There are two kinds of notifications: status notification and event notification. Status notification is used by the gateway to regularly report its status. When the gateway overloads, crashes, or receives outbound packets that cannot be dealt with, event notification will be triggered to notice the controller such events.

The HDR contains communication type and message ID. The IDg payload indicate the gateway from which the message is sent. The NT payload provides the notification message type. When the NT payload is STATUS, which means that the message is used to report gateway status, the DES payload will give a list of gateway status parameters, including CPU utilization, memory utilization, number of tunnels, etc. When the NT payload is set to EVENT, the DES payload indicates the event type and gives detail information about the event. Possible events are OVERLOAD, CRASH or PACKET. PACKET here means that the gateway encounters unknown packet. The RTN payload in response messages is set to ACK, if the notification is accepted and will be responded.

2) The Configuration Message: As is shown in Figure 4, configuration messages are initiated by the controller and responded by the gateway (SDIG device). Controller uses configuration messages to issue configuration elements.

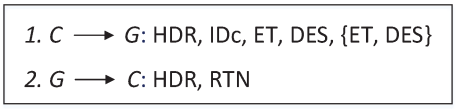


Fig. 4. The Configuration Message

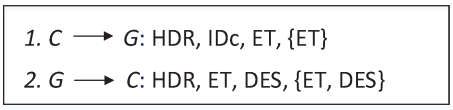


Fig. 5. The Query Message

The HDR contains communication type and message ID. The IDc payload indicates ID of the controller. There can be one or more ET and DES pairs in configuration messages. Each pair provides an element for the gateway. Available values for ET are RULE that means forwarding rule, ACTIVE that means active connection, PASSIVE that means passive connection, and GATEWAY that means instructions for the gateway. If there are more than one elements to be issued, each of them will be delivered through one pair.

The RTN payload in respond messages gives the result of applying the issued elements. SUCCESS will be used while applying successfully, and ERROR followed by error description for failed cases.

3) The Query Message: Figure 5 shows a query message pair that contains a query message from the controller and a response message from the gateway (SDIG device). Controllers uses query messages to get detail information about gateways.

In query messages, the HDR and IDc payload are used same as they are used in configuration messages. There can be one or more ETs in query messages. Each of the ETs specify one element to be queried. In respond messages, query results will be organized in ET and DES pairs each of which corresponds to one ET in query request.

ET payload can take various of values. For example, ADDRESSES for ip addresses of all gateway interfaces, TUNNELS for a IPsec tunnel list, TUNNEL followed by a tunnel ID for details about a certain tunnel.

4) Finite State Machine: Figure 6 depicts the finite state machine of the SDIG device. It illustrates how the customized protocol works. The gateway switches to “gateway waiting” status after started. When it received an unknown packet, it switches to “sending event noti” in which the gateway is trying to send event notification about the unknown packet. When event notification is sent, it switches to “waiting conf” and waits for controller’s reply. After receiving configuration message from controller, it switches to “handling conf” in which the received configuration is applied. If the configuration is applied successfully, it switches to “gateway waiting”.

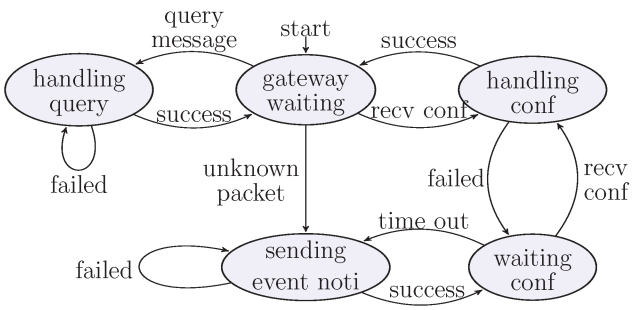


Fig. 6. Finite State Machine

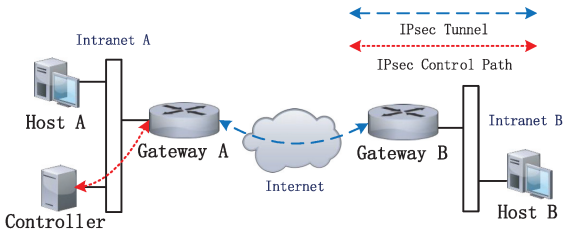


Fig. 7. Experiment Network Topology

Another state "sending status noti" is omitted in the figure. The gateway switches to the "sending status noti" state from "gateway waiting" at regular intervals to send its status information, and switches back when finished.

IV. Implementation and Evaluation

A. Prototype Implementation

We implement a prototype of the system, including an SDIG device and an SDN controller for the SDIG.

The SDIG device is a software IPsec implementation. We develop a gateway agent to perform uniform control, with strongSwan (version 5.3.2) providing basic IPsec functions. In addition, a linux kernel module is developed to deal with unknown packets.

We use OpenDaylight Berryllium SR2 as the controller. A plugin is developed to support our customized southbound protocol as well as IPsec management functions. All IPsec elements are stored in memory to accelerate IPsec decision process. Thread pool is used to gain the ability to handle concurrent requests.

All the programs and data sets used in our experiments can be found in [12].

B. Evaluation

Preliminary experiments are made on the implemented system to analyze the feasibility and effectiveness. The topology is shown in Figure 7. Two hosts with Intel Core i3-4130, 4GB memory and two Gigabit ports are used as gateways. Gateway A runs as SDIG, while Gateway B remains to be a traditional security gateway. Our controller is deployed on a host with Intel Core i5-2400 and 2GB memory. Another two hosts are used to test the connectivity of IPsec tunnel.

1) Tunnel Setup Delay: In our system, the first packet arrived at gateway from protected network triggers the establishment of IPsec tunnel. An important parameter of the system is the delay between arriving of packet and establishment of tunnel. The delay is introduced by our centralized management mechanism. The feasibility of our design will be proved if the delay is acceptable.

We implement a program that sends UDP packet every 0.01 second on Host A. Every packet carries the time when it is sent. Program on Host A will print the exact time when the first packet sets off. Host B is running a program that prints the payload of the first packet it receives from Host A. So Gateway A acts as an initiator and Gateway B is the responder. In this way, total delay can be measured.

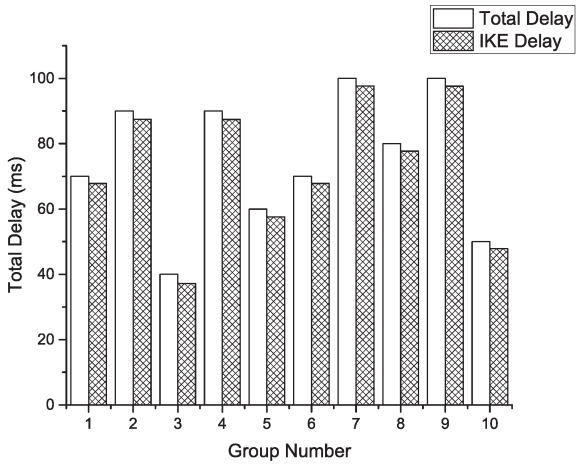
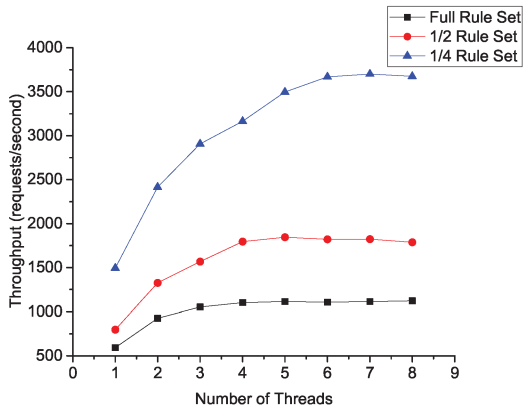


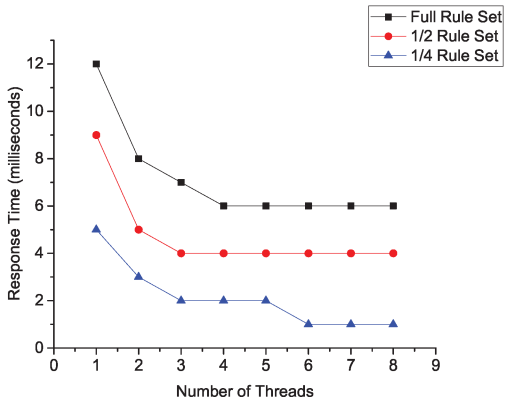
Fig. 8. Tunnel Setup Delays

The total delay consists of two parts: the control delay and the delay caused by IKEv2 negotiation. Wireshark packet capture on Gateway A is used to get the control delay which is introduced by the interval between Notifications and Configuration messages. IKE delay can be calculated by subtracting control delay from total delay.

We use our testing programs to perform 10 tests, results of the experiment are organized and depicted in Figure 8. As the figure shows, total delays are almost equal to the IKE delays. The control delay our system introduces (2.4 ms on average) is much less than the IKE delay (72.6 ms on average). As the experiment in next session shows, the upper bound of the control delay is 12ms which is still much less than the IKE delay. When applied in production environment, the IKE delay may increase sharply depending on the network environment. But the control delay will not change significantly, because the SDIG devices and SDN controllers are always deployed in the same network region. More details about the result can be found at [12].



(a) Throughtput



(b) Response Time

Fig. 9. Experiment Result with Different Numbers of Threads

2) Controller Performance: Another experiment tests the ability of controller to handle unknown packet notifications. More than 65k IPsec rules and the same number of IPsec connections are generated randomly. 8 emulated SDIG devices are used to launch notifications using a test set. The test set consists of 65k source and destination address pairs which correspond to the 65k rules respectively. We run the controller with different numbers of threads, measure the average throughput and response time.

Figure 9(a) shows the throughput with different number of threads, and Figure 9(b) shows the response time. 1/2 (1/4) rule set means that a half (quarter) of the 65k IPsec rules are used during the experiment. The result shows that our controller scales well up to the point where no CPU resources are left (Intel Core i5-2400 has 4 cores and 4 threads). When the number of IPsec rules increases, time consumption of query these rules increases too, which leads to the degradation of controller performance. This is mainly caused by the algorithm controller uses to lookup IPsec rules. In our preliminary implementation, the IPsec rules are queried linearly with an O(n) time complexity. This can be optimized by introducing tree data structure.

V. Related Works

The idea of SDN originates from Clean Slate Program [13] which was launched by Stanford University. Ethane [14] couples extremely simple flow-based Ethernet switches with a centralized controller that manages the admittance and routing of flows, which shows the advantage of using centralized management. Since then, many SDN based solutions have been proposed to solve traditional problems. [15] applies SDN paradigm to mobile wireless network and proposed software-defined wireless network. [16] gives a software-defined access architecture to facilitate interworking in HetNets.

For VPN problems, most solutions are on addressing MPLS VPN management: [17] extended OpenFlow 1.0 to support MPLS, [18] took advantage of SDN to improve the tunnel management performance of VPLS architectures. When talking about IPsec VPN, solutions mainly concentrated on aggregating IPsec flows [19] or enabling OpenFlow switches to provide IPsec service [20]. Our previous work [21] attempted to automatize the configuration of IPsec VPN by using OpenFlow switches.

However, prior works just tried to use SDN based method to solve IPsec problems under current architecture. Improvement of IPsec architecture is not considered. Our work is to reframe IPsec gateways and proposing a new IPsec device.

VI. Discussion and Future Work

In this paper, we propose the SDIG, a new IPsec gateway to achieve net2net IPsec VPN. The proposed system improve IPsec gateway by using SDN controllers and performing centralized management. Centralized management makes the SDIG device easy to manage. Since all the information are collected in the controller and all the commands are issued from the controller, administrators can perform management through monitoring and operating controllers instead of accessing gateways one by one. Also, the centralized control provides a better way to apply high availability and load balancing. Another advantage SDIG brings is that it can integrate IPsec VPN with cloud computing. Using SDN controllers provides a method to virtualize IPsec services, and makes it possible for cloud applications to take advantage of IPsec resources. The result of our preliminary experiments shows that the design is feasible and effective.

However, there are still some problems. When number of rules increases, performance of the controller degrades. We will keep on improving performance of the controller to increase throughput and reduce delay.

Acknowledgment

This work is partly supported by the National High-tech R&D Program (“863” Program) of China (No.2015AA01A202) and research project of Beijing Municipal Education Commission.

REFERENCES

[1] S. Kent and K. Seo, "Security architecture for the internet protocol," Tech. Rep., 2005.
[2] C. Kaufman, P. Hoffman, Y. Nir, P. Eronen, and T. Kivinen, "Internet key exchange protocol version 2 (ikev2)," Tech. Rep., 2014.
[3] O. N. Fundation, "Software-defined networking: The new norm for networks," *ONF White Paper*, 2012.
[4] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.
[5] D. Erickson, "The beacon openflow controller," in *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*. ACM, 2013, pp. 13–18.
[6] "Floodlight openflow controller," http://www.projectfloodlight.org/floodlight/.
[7] "Pox," http://www.noxrepo.org/pox/about-pox/.
[8] "Ryu sdn framework," http://osrg.github.com/ryu/.
[9] P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, B. O'Connor, P. Radoslavov, W. Snow *et al.*, "Onos: towards an open, distributed sdn os," in *Proceedings of the third workshop on Hot topics in software defined networking*. ACM, 2014, pp. 1–6.
[10] J. Medved, R. Varga, A. Tkacik, and K. Gray, "Opendaylight: Towards a model-driven sdn controller architecture," in *Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014*, 2014.
[11] "ipsec.conf: conn reference," https://wiki.strongswan.org/projects/strongswan/wiki/ConnSection.
[12] "sdig-project," https://github.com/linfx7/sdig-project.
[13] N. McKeown and B. Girod, "Clean slate design for the internet," *Whitepaper, Apr*, vol. 14, 2006.
[14] M. Casado, M. J. Freedman, J. Pettit, J. Luo, N. McKeown, and S. Shenker, "Ethane: taking control of the enterprise," in *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 4. ACM, 2007, pp. 1–12.
[15] C. J. Bernardos, A. De La Oliva, P. Serrano, A. Banchs, L. M. Contreras, H. Jin, and J. C. Zúñiga, "An architecture for software defined wireless networking," *IEEE Wireless Communications*, vol. 21, no. 3, pp. 52–61, 2014.
[16] V. Sagar, R. Chandramouli, and K. Subbalakshmi, "Software defined access for hetnets," *IEEE Communications Magazine*, vol. 54, no. 1, pp. 84–89, 2016.
[17] J. Kempf, S. Whyte, J. Ellithorpe, P. Kazemian, M. Haitjema, N. Beheshti, S. Stuart, and H. Green, "Openflow mpls and the open source label switched router," in *Proceedings of the 23rd International Teletraffic Congress*. International Teletraffic Congress, 2011, pp. 8–14.
[18] M. Liyanage, M. Ylianttila, and A. Gurtov, "Improving the tunnel management performance of secure vpls architectures with sdn," in *2016 13th IEEE Annual Consumer Communications & Networking Conference (CCNC)*. IEEE, 2016, pp. 530–536.
[19] V. H. F. Tafreshi, E. Ghazisaeedi, H. Cruickshank, and Z. Sun, "Integrating ipsec within openflow architecture for secure group communication," *ZTECOMMUNICATIONS*, p. 41, 2014.
[20] P. Gorja and R. Kurapati, "Extending open vswitch to l4-l7 service aware openflow switch," in *Advance Computing Conference (IACC), 2014 IEEE International*. IEEE, 2014, pp. 343–347.
[21] Y. Li and J. Mao, "Sdn-based access authentication and automatic configuration for ipsec," in *2015 4th International Conference on Computer Science and Network Technology (ICCSNT)*, vol. 1. IEEE, 2015, pp. 996–999.