DOI:10.3939/j.issn.1673–5188.2014.02.007
http://www.cnki.net/kcms/detail/34.1294.TN.20140604.1151.001.html, published online June 04, 2014

Integrating IPsec within OpenFlow Architecture for Secure Group Communication

Vahid Heydari Fami Tafreshi [1], Ebrahim Ghazisaeedi [2], Haitham Cruickshank [1], and Zhili Sun [1]
(1.Centre for Communication System Research (CCSR), University of Surrey, Guildford, Surrey, GU2 7XH, UK;
2. Department of Systems and Computer Engineering, Carleton University, Ottawa, K1S 5B6, Canada)

Abstract

Network security protocols such as IPsec have been used for many years to ensure robust end-to-end communication and are important in the context of SDN. Despite the widespread installation of IPsec to date, per-packet protection offered by the protocol is not very compatible with OpenFlow and flow-like behavior. OpenFlow architecture cannot aggregate IPsec-ESP flows in transport mode or tunnel mode because layer-3 information is encrypted and therefore unreadable. In this paper, we propose using the Security Parameter Index (SPI) of IPsec within the OpenFlow architecture to identify and direct IPsec flows. This enables IPsec to conform to the packet-based behavior of OpenFlow architecture. In addition, by distinguishing between IPsec flows, the architecture is particularly suited to secure group communication.

Keywords

IPsec; OpenFlow; secure group communication; group domain of interpretation (GDOI); flow-based switching

1 Introduction

As an attempt to embrace the future Internet and its tendency towards software-defined networks (SDN), OpenFlow suggests a move into programmable rather than configurable network deployments. This results in faster innovations through software change rather than infrastructure adaption [1]. OpenFlow works well on the premise that the control plane can be separated from data plane on network packet forwarders and brought into an OpenFlow controller (a server) with centralized network management. All network elements, including routers and switches, are now simple packet forwarders with no complexity. Starting initially with campus networks, data centers such as Google are now extensively reinforced with this evolving architecture [2].

On the other hand, end-to-end security of communication at the IP level is guaranteed by the IPsec framework [3]. As the word "framework" implies, IPsec is not directly limited to any specific security algorithm or technology. Subsequently, the level of security can be tuned by different open standards and combinations to fulfill various immunity requirements of the production environment. Virtual private network (VPN) as a solution for providing a logical channel between two peers over a public and probably insecure network relies on the IPsec for its immunity. Small-office home office (SOHO) scenario or different sites of a corporation which are geographically spread out are other possible use cases to apply VPN remedy over IPsec.

Point-to-point tunnels between two VPN gateways used to be exploited to carry authenticated as well as encrypted traffic from one site to another. However, group domain of interpretation (GDOI) [4]–[6] with IPsec at its core goes even further so that secure communication between various sites called group members (GM) is now possible without any tunnels between these branches.

IPsec as an algorithm-independent framework addresses the confidentiality by encryption as well as the integrity with the aid of hashing as the main security objectives while allows for authenticating the origin of the traffic. Regardless of the core network and its elements, the tunable IPsec protocol with huge install base is simply provisioning the necessary security services for both end entities. Nonetheless, security gained through IPsec is per-packet. This is not deployable to leading future Internet designs such as OpenFlow architecture with flow-based behavior. OpenFlow aims to aggregate different packets into flows and process these flows rather than individual packets. OpenFlow, however, cannot uniquely identify IPsec flows and aggregate/direct these flows accordingly. We provide the ability to distinguish between IPsec flows in order to integrate secure group communication into the OpenFlow architecture. Our ultimate goal is to address this deficiency within OpenFlow by our proposed method. We propose using a security parameter index (SPI) of IPsec within the OpenFlow architecture to uniquely identify and direct IPsec flows.

The rest of the paper is structured as follows. In section 2, we discuss briefly four basic elements used later for our proposed method. This is needed specifically to clarify ambiguities especially those pertaining to a complex protocol like IPsec. The clarification emphasizes characteristics which form the cornerstone of the method in section 3. Subsection 2.1 explores OpenFlow and abstracts the ideas behind this evolving architecture. Subsection 2.2 sanitizes the required features of IPsec itself. The establishment of the secure channel through internet key exchange (IKE) is discussed in 2.3. Subsection 2.4 briefly covers GDOI basics as a cryptographic protocol for group key management for secure group communications. The proposed method is discussed in detail in section 3. Section 4 elaborates on the main use case for the proposed method here which is secure group communication conforming to GDOI standard and its integration into the OpenFlow. Section 5 concludes this paper.

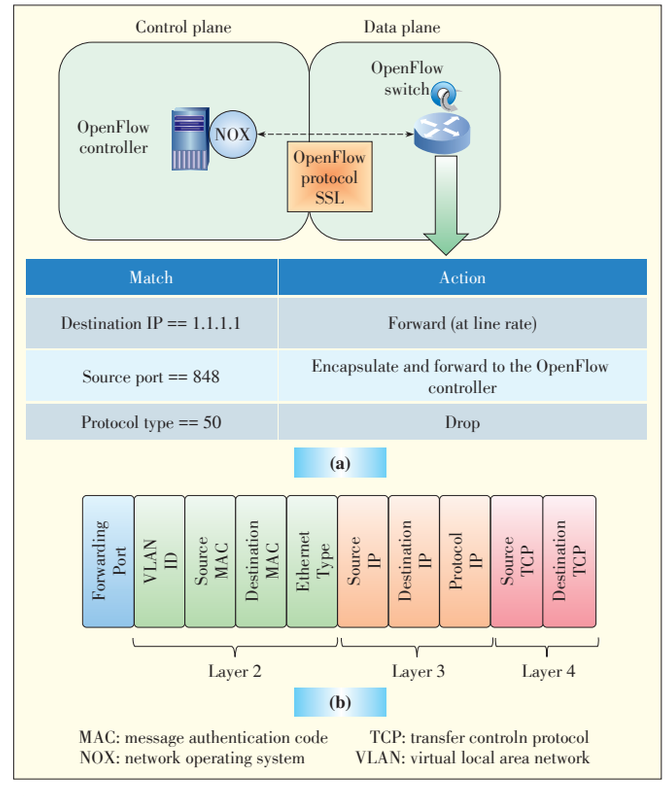
2 Background

2.1 OpenFlow Architecture

OpenFlow improves network programmability and enables packets to be forwarded at a speed approximating the line-rate. This speed is possible due to minimized complexity stemming from the separation of the control plane from data plane [1]. To reduce system complexity, OpenFlow considers all network elements, including routers and switches, as simple, hardware-based packet forwarders. Complexity is thus shifted to the application layer, where software on the OpenFlow controller (a server with sufficient resources) makes various decisions and informs forwarders of the outcomes of these decisions. These outcomes are disseminated as flow-tables across the packet forwarders and define various pairs (match, action). This means that for each incoming packet, if there is a match in the flow-table of the local device, a special action is performed. Three standard actions are Forward, Encapsulate, and Drop. Forward makes the OpenFlow-enabled device act as a router/switch at the line-rate. If no match is found or if the packet is the first in a new, undefined flow, it is encapsulated and forwarded to the OpenFlow controller, where decisions subsequently are made. The packet can also be discarded through a drop action.

The network operating system (NOX) is a programmable interface that facilitates network management by providing an environment for running applications sitting on the OpenFlow controller. The OpenFlow controller communicates with the packet forwarders through the OpenFlow protocol over a secure SSL/TCP channel. With the aid of this open OpenFlow protocol, different routers' and switches' flow-tables can be programmed in a scalable manner. Entries in each flow-table on every OpenFlow packet forwarder are associated with different actions while statistics are being collected. **Fig. 1 (a)** depicts the separation of the control plane from data plane in OpenFlow architecture in addition to the flow-table structure. For instance, if the destination IP address of the incoming packet is equal to 1.1.1.1, the packet is forwarded to a given port. If it has 848 (UDP port for GDOI protocol) as the value for the source port, it will be encapsulated and then forwarded to the controller for further investigation. If the packet is IPsec encapsulating security payload (ESP) packet with type equal to 50, it will be dropped.

The first generation of OpenFlow packet forwarders, called "OpenFlow spec v1.0 conforming switches", defines flow header fields which encompass some features of each incoming packet as illustrated in **Fig. 1 (b)**. When a packet arrives, its header is firstly checked against the Match field and if the header matches any row in the flow-table, the corresponding action is performed. Any combinations amongst these demonstrated 10-tuple can be utilized to define and aggregate flows accordingly. These flow header fields are then exploited in order to specify matches in flow-tables for each incoming packet and perform the corresponding action. However, OpenFlow is currently unable to distinguish between IPsec flows. The authors in [1] emphasize the header fields of "OpenFlow spec v1.0 conforming switch" as the initial and standard header fields with which every OpenFlow switch must comply. This is substantial since later on we introduce our new flow header



▲ **Figure 1. (a) reveals the internal structure of OpenFlow architecture. In (b), flow header fields defined for "OpenFlow spec v1.0 conforming switches" (first generation OpenFlow packet forwarders) are shown. OpenFlow is currently unable to "distinguish between" IPsec flows.**

fields for the OpenFlow interface which is IPsec-aware and also backward compatible to "OpenFlow spec v1.0 conforming switch" header fields.

2.2 IPsec

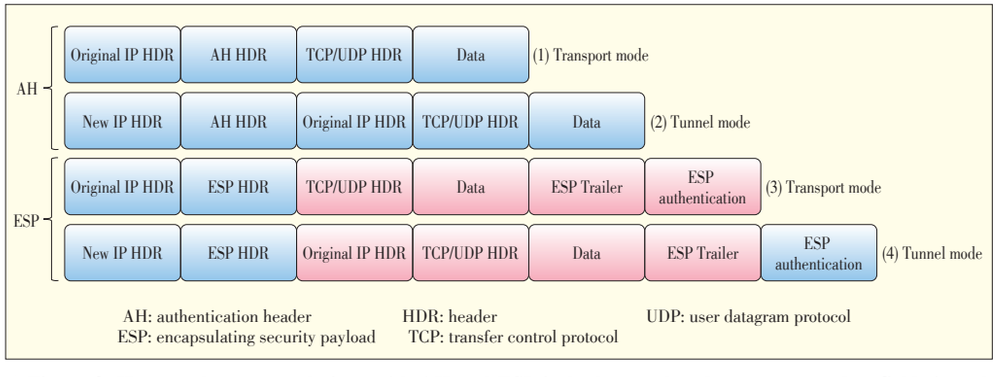
Working at the network layer, IPsec protects the traffic between peers by provisioning encryption as well as authentication from Layer 3 to Layer 7. On the other hand, all the current layer-2 technologies enable the IPsec framework function over them. The IPsec framework comprises five components. Available algorithm choices facilitated for each of these components result in different security solutions with each combination to satisfy various needs.

The first component highlights the IPsec protocol and can support either authentication header (AH) or ESP (protocol type 50 for ESP and 51 for AH).

Each IPsec protocol operates either in transport or tunnel modes. The encapsulations of the IP packet secured by IPsec with AH/ESP in both transport and tunnel operation modes are depicted in **Fig. 2**. Both protocols share provisioning authentication and integrity security services. Nevertheless, confidentiality is not considered in AH. This is crucial to differentiate segments of information which are encrypted and thus unreadable from other readable segments which can be meaningful for the third party in the middle of the conversation (i.e., openflow switch or controller). The second component demonstrates the choice for the encryption/decryption algorithm which pertains to the confidentiality service. As with every cryptographic system, the longer the key, the harder it is for an attacker to break into the IPsec communication. The third component ensures that the IPsec communication is not tampered with in transit and thus provides integrity. The fourth component facilitates authentication of the endpoints in secure communication via IPsec. The last building-block specifies the Diffie-Hellman (DH) algorithm group according to different needs. DH is a public key exchange mechanism that enables both communicating parties to come up with the same key over an unsecured channel. The driven shared key is used by peers for symmetric encryption as well as hashing through message authentication code (MAC) in the second and third components of IPsec, respectively.

In IPsec with AH, a message digest is formed by applying the hashing function to the original IP header and data payload utilizing the shared key. The digest then constructs a new AH header, which is injected into the original packet. The same calculation is performed in the receiving party to find the exact match of hashes. Nonetheless, all the data is transmitted in plaintext. Not considering any encryption mechanism leads to having all the layer-3 information in plaintext and therefore routable. However, the original IP header is also encrypted and unreadable when ESP is in tunnel mode. As we see later, this information in plaintext is immensely valuable for our method to differentiate various IPsec flows. Despite AH, in IPsec with ESP, encryption makes payload and the ultimate transmitters' identifications meaningless to eavesdroppers. Both the IP header and data payload are encrypted in this mode. This is followed by appending a new ESP header (as well as ESP trailer) and ESP authentication fields, including relevant encryption and authentication data to the original packet.

In Fig. 2, IPsec in transport mode merely considers the encapsulation of the data payload and transfer control protocol (TCP)/user datagram protocol (UDP) data (layer-4 and above). Nonetheless, tunnel mode suggests that the whole IP datagram is encapsulated within a new IP packet. On the other hand, while secure communication between gateways demands tunnel mode of the IPsec solution, transport mode facilitates host-to-host immune transmissions. While in AH, the original IP header remains unencrypted, and thus leaves the routing intact, the original TCP/UDP header is encrypted in ESP. Both protocols in different modes have their SPI in plaintext within the ESP/AH header. SPI differentiates various ongoing conversations at the receiving party.



▲**Figure 2. IPsec packet encapsulations with AH and ESP in both tunnel and transport modes; fields in red are encrypted and thus known only to end entities (i.e., not any third party in the middle of conversation including OpenFlow switch or controller). It is also noteworthy that both protocols in different modes have their SPI in plaintext within the ESP/AH header.**

2.3 Internet Key Exchange

The key exchange mechanism in IPsec is accomplished through IKE version 2 protocol [7]. The key exchange process with IKE finally leads to the construction of security association (SA) for IPsec. To establish an IPsec connection, IKE involves two phases. During these phases a set of messages is communicated, either in main mode or aggressive mode, resulting in the establishment of a secure channel between the peers. Phase 1 enables peers to agree on the security proposals generally as

well as the shared secret key and authenticate each other. Upon finalizing a secure tunnel in phase 1, phase 2 negotiates the custom security parameters between peers. On completion of phase 2, an SA is formed in a unidirectional manner. Each SA, as a logical connection, defines the way that the traversing traffic will be processed. Subsequently, the same security processing applies to the traffic associated with every SA.

Because a single SA specifies only two parties in a unidirectional manner, each party holds a security association data base (SADB) comprising multiple SAs, where each SA is associated with a different peer. SPI comprises an arbitrary 32-bit value utilized by a receiving party to differentiate the SA to which an incoming IPsec packet is associated. For a unicast communication, SPI on its own can specify an SA. Other parameters, such as the type of IPsec protocol can come along with SPI to highlight a unique SA. However, [8] emphasizes that the sufficiency of SPI on its own to determine an individual SA to which inbound traffic will be mapped or necessity to exploit other parameters in conjunction with SPI is a local matter. As we will see later, SPI can fall into a domain large enough to uniquely identify an SA. The following tuple illustrates the parameters any combination of which can be used to construct the primary key for SADB locally:

{SPI, IPsec Protocol Type (AH/ESP), Peer IP Address, Transform Set, Secret Key, SA Lifetime}

A combination of the elements in the vector above will shape SADB and determine various SAs stored on each peer.

2.4 GDOI

Group Encrypted Transport VPN solution[6], [9]–[10] with GDOI its heart is deemed to provide revolutionary and ultimate technology that reduces complexity and overheads pertaining to the need for scalable as well as secure transport remedy for always-on and dynamic connectivity of extremely integrated network sites spread over diverged domains. Any-to-any network connectivity is guaranteed to be end-to-end encrypted, authenticated and globally scalable for all applications namely voice, video and data with both unicast as well as multicast traffic. In other words, with the advent of GDOI architecture, the arduous obstacle of complexity pertaining to manageable as well as scalable VPN solutions for an abundance of fully-meshed sites (not only two endpoints) is not out of the question anymore[11].

GDOI as a cryptographic protocol for key management is based on IKE. While IKE ensures pairwise security associations between various peers, GDOI utilizing IKE phase 1 between each GMand a key server (KS) ends up with a single and common SA between all the GMs. Additional to pair-wise SAs with IKE phase 1, GDOI also "interprets" IKE to come up with a single SA for the group security domain. In other words, as the foundation of the GET VPN solution, GDOI defines IKE Domain of Interpretation (DOI). Utilizing UDP port 848, GDOI messages create, delete, and maintain SAs established between authenticated and authorized GMs. KS rekeys the group before current keys downloaded at the time of registration by GMs expire. As **Fig. 3** reveals, regardless of what the core network's technology is (WAN, MPLS, OpenFlow, etc.), each GM initially exchanges a GDOI Register message with KS which leads to downloading required keys and policies via bidirectional arrows. KS at some point in time before current keys expire pushes Rekey message which entails new policies as well as keys to given GMs via unidirectional arrows. In this way, encrypted multicast/unicast conduits are established amongst all GMs, not merely two endpoints, to communicate without any tunneling in place.

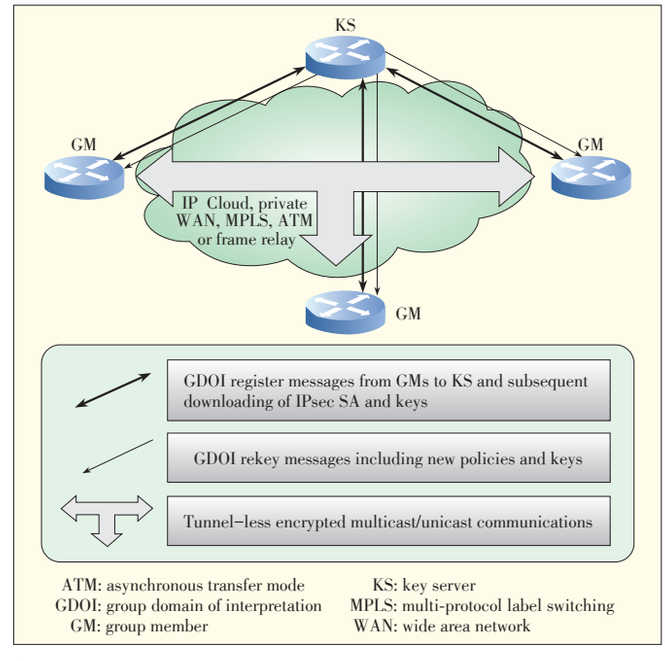
Tunnel-less but secure communication with GDOI for Transport VPN requires GMs to first dispatch registration queries to KS. With the aid of GDOI, KS authenticates and authorizes the given GM and sends back keying materials in addition to the IPsec policy needed for secure GM-to-GM(s) unicasting/ multicasting back to the given GM.

3 Proposed Method and Discussion

3.1 Integration of IPsec within OpenFlow Architecture

Increased control gained through custom forwarding of OpenFlow does enable different flows to be processed in different ways. OpenFlow is advantageous from this wide range of definitions for flows of any combination of header field defined for "OpenFlow spec v1.0 conforming switch" in section 2.

A can highlight a flow. However, when it comes to end-to-



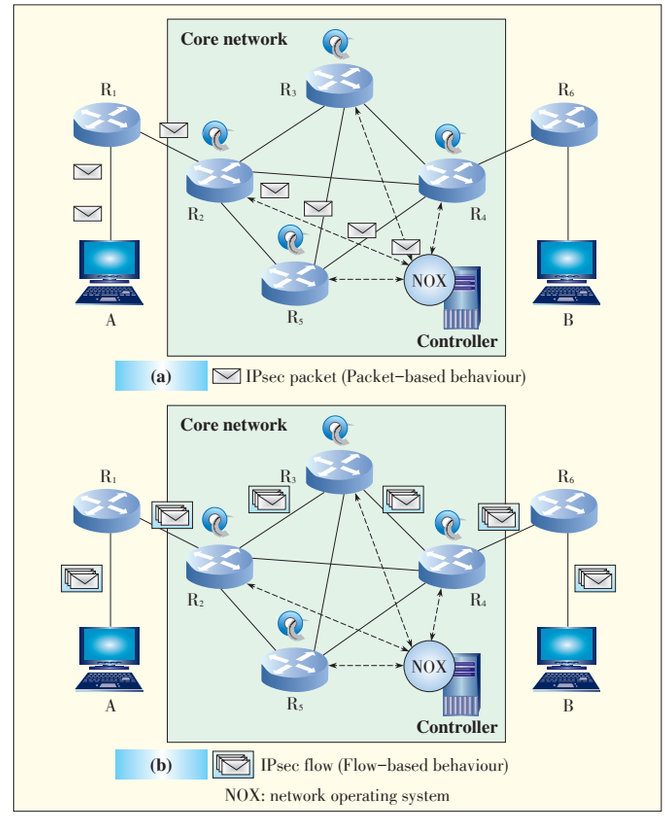
▲ **Figure 3. Upon downloading IPsec policies and keys from KS, GM is now registered with the "IPsec SA for the group" and can exchange unicast/multicast traffic securely with other GMs laying away the KS.**

end IPsec transmission, OpenFlow is unable to detect encrypted IPsec headers, which is discussed in section 2.2, and thus cannot aggregate them into a flow. The only exception is when OpenFlow filters the incoming packets to find a match for the IPsec protocol type, which is not sufficient to uniquely identify a flow because various but irrelevant entities might disseminate IPsec traffic for each other. Encrypted packet headers in IPsec act as a deterrent so OpenFlow switch treats them as a distinct flow [12]. With the standard header fields of "OpenFlow spec v1.0 conforming switch" today, the IPsec ESP-encrypted packets cannot be processed based on layer 4 and above information in both transport and tunnel modes. OpenFlow architecture also cannot deal with decrypting layer-3 information for IPsec with ESP in tunnel mode if any boundary packet forwarder peels the new IP header off before processing further for original layer 3 discovery and then delivery (when VPN tunnel terminates one hop before). To sum up, OpenFlow architecture is unable to aggregate flows of IPsec with ESP in both transport and tunnel modes because layer-3 and above information is encrypted and therefore unreadable for OpenFlow interface. Our method tries to find the distinguishing factor for uniquely identifying IPsec flows and directing these flows accordingly in order to replace the packet-based behavior of OpenFlow architecture towards IPsec with flow-based behavior. We argue that through our proposed method, in the OpenFlow environment we can overcome the abovementioned obstacles in the core network.

Fig. 4(a) shows the baseline scenario in which A tries to establish a secure communication with B via IPsec. It is possible that A acts as a remote access server which serves many clients or shares files with them via IPsec communication (can be KS in GDOI-like implementation). R_2, R_3, R_4 and R_5 form the core network elements in which OpenFlow architecture is employed. R_1 and R_6 can be thought of as security-aware gateways between which IPsec tunnel mode is constructed. In the transport mode of IPsec, they can be seen as local routers while end hosts address immune communication directly.

The dashed arrows indicate the conduits for the OpenFlow controller to securely talk to OpenFlow switches across the core network by OpenFlow protocol. Without our proposed method, IPsec packets from endpoint A to B in the figure reaching R_2 cannot be treated as a flow and should be sent to the OpenFlow controller one by one for decision-making if they are encrypted with ESP (unreadable layer 3 and above information). This will degrade network performance and impose a huge processing burden on the OpenFlow controller within the core network. This is because each IPsec packet is treated with packet-based behavior by being encapsulated and sent to the OpenFlow controller for decision-making one by one. Our goal is to aggregate IPsec packets associated with each secure communication and forward them as flow satisfying arbitrary routing policies of the core network for instance. This might be the case if in an attempt to assign a specific physical route which highly considers security countermeasures and thus is more trustworthy for the IPsec communications (or other traffic engineering tasks such as seeking more available bandwidth), IPsec flows are separated from other flows and then forwarded through this route. Another use case as we will discuss is when more than two endpoints as group members participate in secure group communications over IPsec via GDOI.

Fig. 4(b) shows that R_2 through our method will eventually separate IPsec flow from other incoming traffic sent by R_1, such as http, and direct it via capable and highly trustworthy R_2-to-R_3-to-R_4 links to R_6 as the egress point. In packet-based behavior of OpenFlow architecture, encrypted packets must be encapsulated and then traverse the OpenFlow controller one by one for further processing. Nevertheless, we aim to aggregate IPsec traffic at R_2 and treat it as a flow without involving the OpenFlow controller's resources for processing each packet individually. Specifically, while the problem was that when packets are encrypted using ESP, the flow identifiers are encrypted and hence cannot be used to distinguish flows, we propose using the SPI of IPsec within the OpenFlow architecture as the distinguishing factor for uniquely identifying IPsec flows



▲**Figure 4. (a): Each IPsec packet is treated with packet-based behavior by being encapsulated and sent to the OpenFlow controller for further decision making one by one. (b): Flow-based behaviour through our proposed method, aggregation of given IPsec traffic along with its separation from other IPsec traffic in the core network have been accomplished.**

and directing these flows accordingly.

The functionality of our design is irrespective of IPsec modes or protocols. This makes the remedy flexible enough to cope with all four different encapsulations (Fig. 2). However, because the security database (SDB) construction on the OpenFlow controller is slightly different in transport mode than in tunnel mode, we bring two scenarios here for different modes. The design needs to consider the fact that network elements in the core network are simple packet-forwarders that are security-unaware (backward compatible to "OpenFlow spec v1.0 conforming switches"). In other words, we cannot expect any cryptographic processing on these OpenFlow switches. They are only capable of finding a simple match for each incoming packet against their flow-table and taking a particular action, like forwarding, and subsequent packets of the same match accordingly to treat them as a flow.

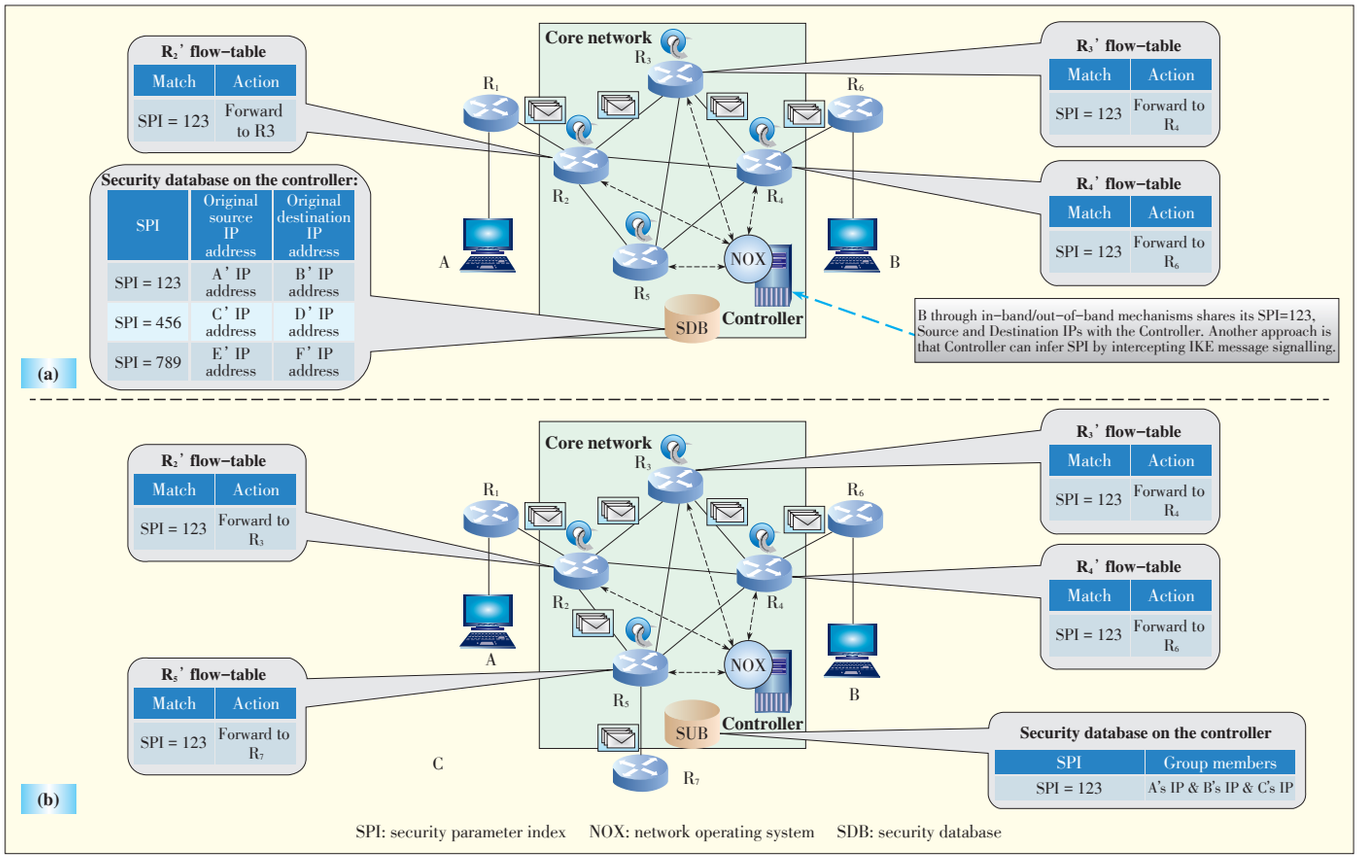
However, this flexibility acquired through the simplicity of OpenFlow architecture cannot distinguish "between" IPsec flows, which is now needed to adapt to secure group communication in GDOI-like architecture for instance. This is due to the fact that the distinguishing factor (if residing in layer-3 or above) is encrypted in the ESP protocol. Each incoming packet encrypted by IPsec with ESP needs to be forwarded to the OpenFlow controller if any information above the IP layer is required for flow-table match-finding.

3.2 Considerations for ESP in Transport Mode

In section 2.3, the first set of messaging between end-devices forms the secure channel over which the transmitters communicate. Once the agreement by end-devices has been reached (IKE phase 2 finished), SAs are established separately for each direction by A as well as B and stored locally in their SADBs. Here, we consider IKE negotiations between endpoints irrespective of the proposed method because SAs need to be constructed prior to treating secure IPsec communication as a flow. Once SAs are established via IKE, the first IP datagram containing the actual secure data onwards can be handled with the proposed design as a flow. Finding a match for header fields listed in Fig. 1b for IPsec on an OpenFlow switch and forwarding based on that fails because end-to-end secure communication ensures that the transmission is unreadable to any entity in the middle when it is ESP for layer-3 and above [12]. On the other hand, these fields are considered as assets accessible only to end-entities who might be reluctant to share them with third parties. The OpenFlow controller initially determines each flow with the aid of the first packet of the communication. This is reasonable because in the beginning, the flow-table has no entry of the flow information before launching the communication. Nevertheless, for IPsec flows, the relevant information is an asset (secret) and thus only both ends have access to it. Because SAs are formed in each direction, each end device is responsible for sharing the required information (here SPI) with the OpenFlow controller prior to traversing the actual flows. Another approach is to let the controller itself infer the SPI because the controller can intercept all IPsec session setup traffic and learn the SPI used between hosts. The SPI acts like a cookie for IPsec where for A-to-B secure communication (two ends, not a group), B firstly determines the SPI value for A-to-B SA and announces it via IKE to A who then carries it in its header field (either AH or ESP header) of IPsec packet(s) to B in plaintext. Consequently, in theory, either A as the data originator should share the received SPI specified by B to the OpenFlow controller or the controller itself infers it directly by intercepting IKE messages.

Upon establishing each SA, the end-device populates its SADB table locally with the relevant security related information. So far, only A and B in Fig. 4 are aware of the security credentials pertaining to IPsec communication between one another. Each SA in each direction can be associated with an SPI number. Subsequently, 2^{32} different SAs can theoretically be established and differentiated between two end-hosts on each site. The SPI is the same for different sequence numbers of the same IPsec communication in a unidirectional manner, and this makes it an appropriate candidate as well as a distinguishing factor among various flow header fields in the design (with more than two entities, SPI also remains the same within a group domain in GDOI). To sanitize it more, bear in mind that IPsec is an immune communication from one sender to another receiver in a one-way direction in which the relevant SA is associated with an SPI carried within AH/ESP headers in plaintext. As a result, for the receiving party, this SPI determines the corresponding SA and thus how the IPsec packet (and resultant flow) will be processed based on the security policy already agreed on mutually via IKE.

Back to **Fig. 5(a)**, we suggest that B shares the SPI with the OpenFlow controller either through in-band (if controller intercepts IKE messages and infers SPI base on them) or out-of-band channels for secure transmission A-to-B before disseminating the actual data. A might have big data and be willing to transmit it in a secure manner to B for instance. The dashed blue arrow reveals the process of handing out the SPI to the OpenFlow controller. Our method requires a SDB on the OpenFlow controller. This SDB contains security related information for IPsec communications. The amount of security credentials shared with the OpenFlow controller is in the end-host's hands. However, our design emphasizes that for flow-based behaviour towards IPsec within OpenFlow architecture, SDB should be populated with SPI at least. In IPsec transport mode, original layer 3 information is also added. Upon sharing SPI with OpenFlow controller by B, the OpenFlow controller must perform an existence check against SDB looking for the announced SPI. If duplicated SPI coexists, the OpenFlow controller should use original layer 3 information as complementary to SPI to uniquely identify the IPsec conversation and update the packet forwarders on the way accordingly. Next, we introduce our new flow header fields for OpenFlow interfaces on the



▲**Figure 5.(a) End-entity B shares its SPI with the controller through in-band/out-of-band mechanisms to add the ability to 'distinguish between' IPsec flows. (b) A, B & C form a group for secure communication in a multicast-like manner, from A to both B and C for instance, based on GDOI.**

switches which contain the new field "SPI" in tuple below in addition to that already mentioned Fig. 1b: {Forwarding Port, VLAN ID, Source MAC, Destination MAC, Ethernet Type, Source IP, Destination IP, IP Protocol, Source TCP, Destination TCP, SPI}

The SPI in plaintext is carried within AH/ESP headers. Therefore, OpenFlow switches are able to detect it directly. The addition of the SPI header field is backward compatible with OpenFlow spec v1.0 conforming switches and does not deem that network elements have any cryptographic capabilities and thus is scalable at the minimum cost.

In a similar way to Fig. 5(a), with **Fig. 5(b)**, A, B and C form a group for secure communication in multicast from A to both B and C with the same method in a GDOI-like manner. The group is associated with SPI = 123 and OpenFlow forwarders are updated accordingly. R_2 now forwards the incoming packets with SPI = 123 to both R_3 and R_5 to form the IPsec flow for the group under the common SA.

3.3 Considerations for ESP in Tunnel Mode

The main difference is that in tunnel mode the original layer 3 information is itself encrypted. Consequently, the OpenFlow controller stores new IP source and destination information in addition to SPI within its SDB at the minimum. This information is needed in case the same SPI has been already installed within SDB and thus more information is required to uniquely identify an IPsec flow. In our scenario, the OpenFlow controller now makes the decision to forward IPsec flows fulfilling its local routing policy and goals by updating appropriate switches' flow-tables while the end to end security is still guaranteed. However, in addition to other header fields, SPI will now also be included for determination of IPsec flows.

To sum up, as the flow header fields defined for OpenFlow spec v1.0 conforming switches indicate in Fig. 1(b), some original flow identification information such as TCP/UDP headers become unavailable with IPsec ESP encrypted traffic for in-path OpenFlow switches to identify/distinguish. With the aid of SPI, which is unencrypted but authenticated in ESP Tunnel Mode, for example, we propose that in-path OpenFlow switches should not only read SPI but can also differentiate IPsec flows accordingly. Using SPI information for classification requires the architecture to embed a mechanism to notify the controller of updates on the SPI values through either in-band or out-of-band mechanisms, such as interpreting IKE negotiations (this can be done prior to actual end-to-end secure communication or through interpreting the first packets of a given IPsec

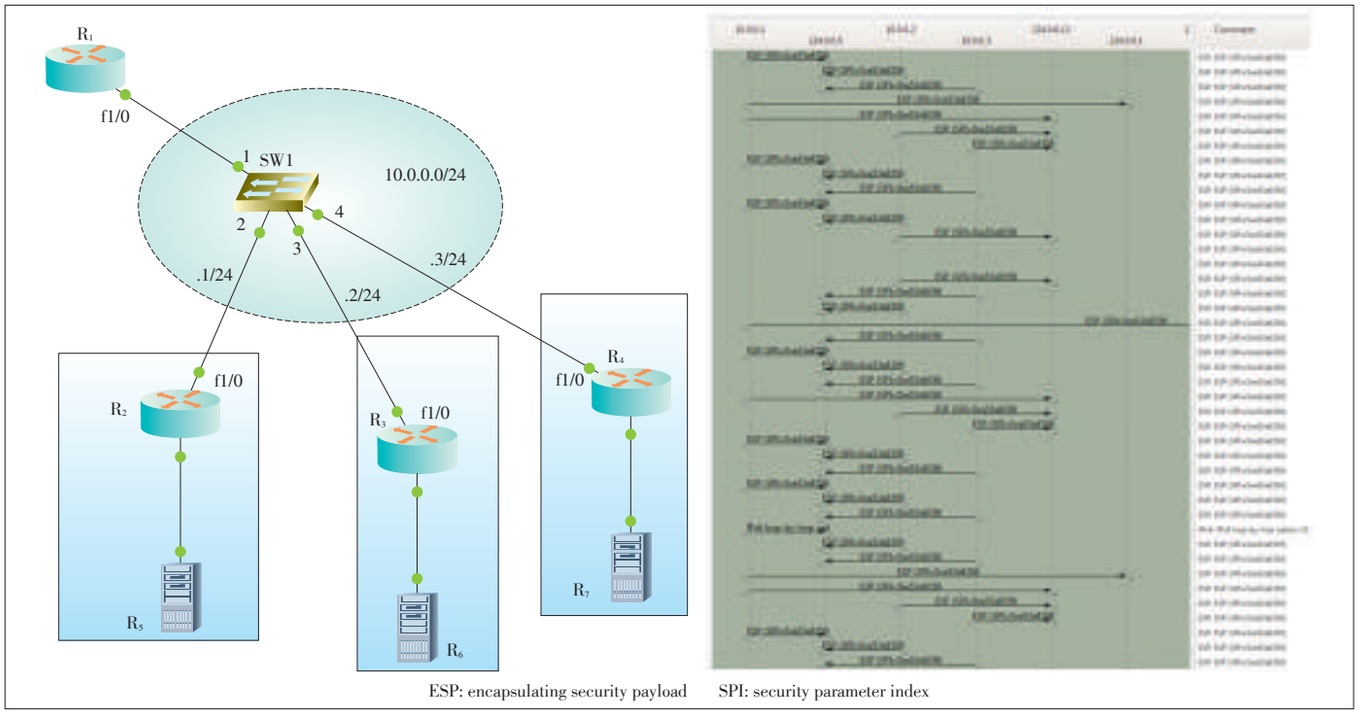
flow by the controller). No matter which, the controller using this mechanism needs to update in-path OpenFlow switches of the given SPI so that it can be read and interpreted for the broad range of intended flow definitions, presuming that our new flow header fields are in place for OpenFlow interfaces on the switches which contain the new field called "SPI" (in addition to the ones defined for OpenFlow spec v1.0 conforming switches displayed in Fig. 1(b)).

Besides the merits achieved by integrating IPsec flows into OpenFlow architecture such as secure group communications based on GDOI standard as discussed in section 4, classifying/policing/shaping IPsec flows let us meet different end-to-end QoS goals in networks as [12] also points out. For instance, usage of SPI within our proposal here enables the OpenFlow switch to perform class-based queuing (CBQ) whereby cryptographically protected traffic among different applications, users (or groups), user affiliations and so forth can be distinguished with an improved level of granularity. Remember that CBQ also accomplishes priority queuing where the preference with which the flows are serviced (can be reliant on service level agreements (SLAs) between different domains for instance) as well as the amount of the queued traffic for them are determined.

4 Use Case: Secure Group Communication Based on GDOI

Traditionally, point-to-point tunnels between VPN gateways were used to carry authenticated and encrypted traffic from one site to another (two ends). For secure group communication with GDOI, encryption/authentication is separated from transport. The merit of this is that secure communication between various sites (more than two) is possible without any tunnels between these branches. This does open the door also for the OpenFlow architecture in the core network eliminating any need for crypto functionality to address transport requirements.

In **Fig. 6** (left), we tried to emulate through Cisco infrastructure [13] GDOI between three nodes, namely, R_2, R_3 and R_4 (can be thought of as A, B and C in Fig. 5(b)) as the GMs. R2, R_3 and R_4 with assigned IP 10.0.0.1/24, 10.0.0.2/24 and 10.0.0.3/24 (all on one subnet), respectively, form a group looking for secure communications through GDOI. R1 will play the role of KS in there. It is likely that the OpenFlow controller serves as the KS. SW1 will represent the core network, which is OpenFlow equipped with our method to respect distinct IPsec flows. GDOI can operate over all the core technologies and therefore must remain infrastructure-independent. The objective here is to eavesdrop on the SW1 after proper GDOI implementation between R_2, R_3 and R_4 via Wireshark to infer the SPI associated with this group domain. Wireshark Flow Graph (Fig. 6). captures all the encrypted communications on the subnet within the group (10.0.0.0/24) after GDOI implementation showing that all the group members share the same SPI for IPsec ESP for the group domain communications. SW1 is required to respect our method through the ability to "distinguish between" IPsec flows using SPI in order to integrate the



▲**Figure 6. Left: baseline scenario in GNS3; R_2, R_3 and R_4 are willing to form a group based on GDOI. Right: Wireshark Flow Graph highlights the captured SPI. The same SPI is used amongst all the group members for secure group communications after proper GDOI implementation.**

notion of secure group communication within SDN. Despite AH, in IPsec with ESP, encryption makes payload and the ultimate transmitters' identifications meaningless to the eavesdroppers. This highlights the main use case for our method within OpenFlow. R_2, R_3 and R_4 were already coded for multicast OSPF as well as PIM to generate some multicast traffic before and after GDOI implementation to highlight the role of this IPsec-based group control protocol. As Fig. 6 (right) reveals, upon finishing GDOI implementation, all the communications originating from GMs (R_2, R_3 and R_4) destined for any multicast address including 224.0.0.5 (for multicast OSPF) or 224.0.0.13 (for PIM multicast) are secured with IPsec ESP while all the communication within this group domain is sharing the same SPI.

5 Conclusion and Future Work

In this paper, we have addressed the deficiency for interworking of OpenFlow with IPsec in both IPsec tunnel as well as transport modes. OpenFlow architecture cannot aggregate flows of IPsec with ESP because layer-3 and above information is encrypted and therefore unreadable. In this paper, we have proposed using the SPI of IPsec within the OpenFlow architecture in order to uniquely identify IPsec flows and direct these flows accordingly. This replaces packet-based behavior of OpenFlow architecture towards IPsec with a flow-based behavior and removes the obstacle of encrypted flow identifiers. We also proposed new flow header fields for OpenFlow switches/interfaces which contain SPI for switching IPsec flows. Sharing SPI with the OpenFlow controller will not jeopardize the immunity of end-to-end IPsec conversation because they are already in plaintext. The proposed method facilitates the ability to distinguish between IPsec flows in order to integrate secure group communication into the OpenFlow architecture. The main use case where identifying "between" IPsec flows can be useful is when secure group communication is required in a similar way to GDOI architecture as discussed.

We will carry out further works on simulating the proposed method in order to evaluate its scalability as well as the performance in the next step based on [14].

References

[1] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, pp. 69–74, 2008. doi: 10.1145/1355734.1355746.
[2] H. Hu, J. Bi, T. Feng, S. Wang, P. Lin, and Y. Wang, "A Survey on New Architecture Design of Internet," in *Computational and Information Sciences (ICCIS), 2011 International Conference on*, Chengdu, China, 2011, pp. 729–732. doi: 10.1109/ICCIS.2011.57.
[3] *Security Architecture for the Internet Protocol*, RFC 4301 (Proposed Standard), 2005.
[4] *The Group Domain of Interpretation*, RFC 6407 (Proposed Standard), 2011.
[5] *The Group Domain of Interpretation*, RFC 3547 (Proposed Standard), 2003.
[6] Y. Bhaiji, *Network security technologies and solutions*. Indianapolis, IN: Cisco Press, 2008.
[7] *Internet Key Exchange Protocol Version 2 (IKEv2)*, RFC 5996 (Proposed Standard), 2010.
[8] *IP Encapsulating Security Payload (ESP)*, RFC 4303 (Proposed Standard), 2005.
[9] S. Wilkins and F. H. S. III, *CCNP security SECURE 642–637 : official Cert guide (master CCNP SECURE 642–637 exam topics ; assess your knowledge with chapter-opening quizzes ; review key concepts with exam preparation tasks ; practice with realistic exam questions on the CD-ROM)*. Indianapolis, Ind: Cisco Press, 2011.
[10] K. Hutton, M. Schofield, and D. Teare, *Authorized self-study guide : Designing Cisco network service architectures (ARCH)*. Indianapolis, IN: Cisco Press, 2009.
[11] *CCIE security practice labs*. Indianapolis, Ind: Cisco Press, 2004.
[12] V. Fineberg, "A practical architecture for implementing end-to-end QoS in an IP network," *Communications Magazine, IEEE*, vol. 40, no. 1 pp. 122–130, 2002. doi: 10.1109/35.978059.
[13] *Graphical Network Simulator*[Online]. Available: http://www.gns3.net/
[14] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: rapid prototyping for software-defined networks," in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, Monterey, California, 2010. doi: 10.1145/1868447.1868466

Manuscript received: 2014-01-25

Biographies

Vahid Heydari Fami Tafreshi (v.fami@surrey.ac.uk) received his BSc in computer software engineering from Shomal Higher Education Institute, Iran, in 2007. He received his Cisco Certified Network Associate (CCNA) and Cisco Certified Network Associate Security (CCNA-Security) certificates from the Cisco Academy at London Metropolitan University, UK, in 2009. He received his MSc in internet computing from the University of Surrey, UK, in 2010. He is currently working and pursuing a PhD degree at the Centre for Communication Systems Research (CCSR), Department of Electronic Engineering, University of Surrey, UK. His main research interests include internet protocols and architecture, network security and multicasting.

Ebrahim Ghazisaeedi (eghazisaeedi@sce.carleton.ca) received his MSc degree in Mobile and satellite communications from the University of Surrey, UK, in 2011. He is currently pursuing a PhD degree in electrical and computer engineering at the Department of Systems and Computer Engineering, Carleton University, Canada. His main research interests are in communication networks, network virtualization, and network optimization.

Haitham Cruickshank (h.cruickshank@surrey.ac.uk) is a senior lecturer at the University of Surrey. He has worked there since January 1996 on several European research projects in the ACTS, ESPRIT, TEN-TELECOM and IST programmes. His main research interests are network security, satellite network architectures, VoIP and IP conferencing over satellites. He also teaches data and Internet networking and satellite communication courses at the University of Surrey. He is a member of the Satellite and Space Communications Committee of the IEEE ComSoc and a chartered engineer and corporate member of the IEE in the UK.

Zhili Sun(z.sun@surrey.ac.uk), Chair of Communication Networking, has been with the Centre for Communication Systems Research (CCSR), Department of Electronic Engineering, Faculty of Engineering and Physical Sciences, University of Surrey since 1993. He got his BSc in Mathematics from Nanjing University, China, in 1982, and PhD in Computer Science from Lancaster University, UK, in 1991. He worked as a postdoctoral research fellow with Queen Mary University of London from 1989 to 1993. He has been principle investigator and technical co-coordinator in many projects within the EU framework programs, ESA, EPSRC and industries, and has published over 125 papers in international journals, book chapters and conferences. He has published a book as sole author titled "satellite networking–principles and protocols" by Wiley in 2005, a book as contributing editors of "IP networking over next generation satellite systems" published by Springer in 2008, and another book as contributing editor to the 5th edition of the text book "Satellite Communications Systems–systems, techniques and technology" published by Wiley in December 2009. His research interests include wireless and sensor networks, satellite communications, mobile operating systems, traffic engineering, Internet protocols and architecture, QoS, multicast and security.