Hindawi Publishing Corporation
Mobile Information Systems
Volume 2016, Article ID 1708970, 12 pages
http://dx.doi.org/10.1155/2016/1708970

Research Article

Securing SDN Southbound and Data Plane Communication with IBC

JunHuy Lam, Sang-Gon Lee, Hoon-Jae Lee, and Yustus Eko Oktian

Department of Ubiquitous IT, Division of Computer & Information Engineering, Dongseo University, Busan 617-716, Republic of Korea

Correspondence should be addressed to Sang-Gon Lee; nok60@dongseo.ac.kr

Received 21 March 2016; Revised 22 June 2016; Accepted 4 July 2016

Academic Editor: Juan C. Cano

In software-defined network (SDN), the southbound protocol defines the communication between the control plane and the data plane. The agreed protocol, OpenFlow, suggests securing the southbound communication with Transport Layer Security (TLS). However, most current SDN projects do not implement the security segment, with only a few exceptions such as OpenDayLight, HP VAN SDN, and ONOS implementing TLS in the southbound communication. From the telecommunication providers' perspective, one of the major SDN consumers besides data centers, the data plane becomes much more complicated with the addition of wireless data plane as it involves numerous wireless technologies. Therefore, the complicated resource management along with the security of such a data plane can hinder the migration to SDN. In this paper, we propose securing the distributed SDN communication with a multidomain capable Identity-Based Cryptography (IBC) protocol, particularly for the southbound and wireless data plane communication. We also analyze the TLS-secured Message Queuing Telemetry Transport (MQTT) message exchanges to find out the possible bandwidth saved with IBC.

1. Introduction

Software-defined network (SDN) is a new network technology that separates the intelligence of the network by decoupling the control and data planes. In order to achieve that, a new network element was introduced into the network, the SDN controller. It centralizes the network control plane, manages the network data plane, and provides the platform that eases the development of the management plane or, in other words, the network applications. The network switches that take on the role of the network's data plane become forwarding devices in SDN; they forward the packets in accordance with the flow tables received from the SDN controller unquestioningly.

SDN introduces three new protocols into the network, namely, the northbound protocol, east/west-bound protocol, and southbound protocol. The conceptual view of SDN with both the wired and wireless data planes is as shown in Figure 1.

The northbound protocol is used by the management plane or network applications to communicate with the control plane (the SDN controllers) to perform tasks such as load balancing via load adaption [1] and Quality of Service (QoS) [2]. The security risks and requirements of the northbound communication are dependent on the network application.

In the SDN environment, security applications or tools can also be used to provide network security from the management plane. SDN allows applications to monitor the network traffic and have a network-wide view. Hence, identity revocation can be carried out easily with applications or tools that detect malicious nodes such as the intrusion detection system (IDS) [3], Distributed Denial-of-Service (DDoS) detection [4], and network monitoring [5, 6].

The east/west-bound protocol is used for the communication within the control plane or, specifically, the communication between the SDN controllers and the data stores. Unfortunately, the SDN controllers currently available are vendor specific as those of the time of writing. They have neither the agreed east/west-bound protocols nor the security for them, with Open Network Operating System (ONOS) being the exception [7]. However, the security of this communication is especially important for the distributed SDN. It ensures that no malicious controllers are snooping for network information or even driving the network.

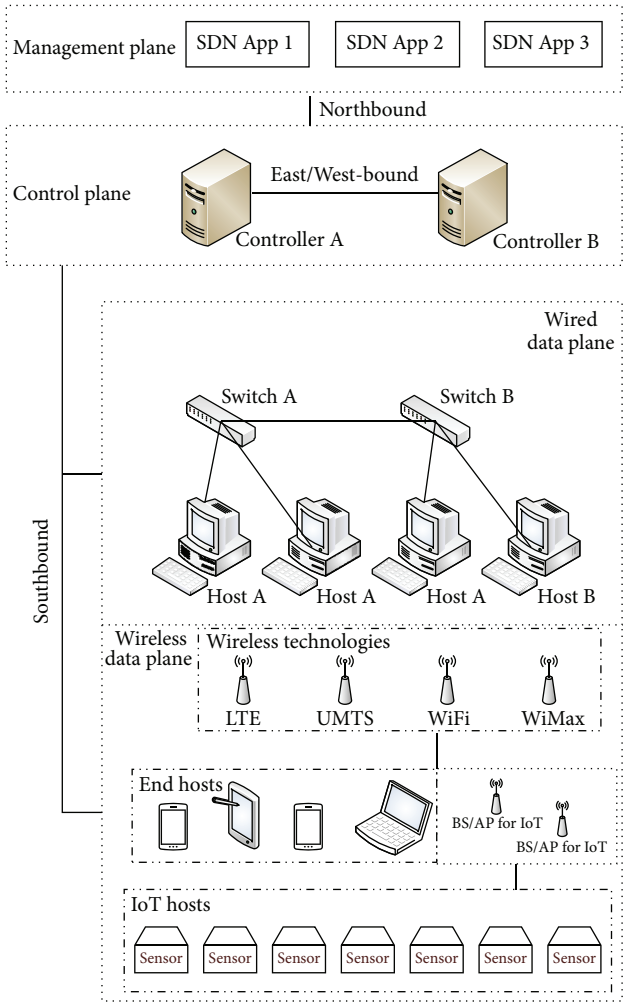

Figure 1: The conceptual view of SDN with both wired and wireless data planes.

In the single controller SDN implementation, a compromise of the controller will enable the attacker to control the entire network. However, in the distributed SDN, the network is spread across all the available controllers within the network. In order to minimize the effect following the compromise of a single controller in the distributed SDN, the east/west-bound communication has to be protected. If it is left unprotected, the malicious controller will have the ability to manipulate all other controllers in the entire network.

The attacker can also insert a malicious data store into the network to obtain a duplicate or backup copy of the network information from other data stores within the network. By utilizing this information, the attacker will then be able to learn the network topology and carry out relevant attacks accordingly. Worse still, the attacker can inject the desired flow through a malicious data store.

The southbound protocol defines the communication between the control plane and the data plane. There is a commonly agreed protocol for the southbound communication which is the OpenFlow protocol [8], standardized by the Open Networking Foundation (ONF). However, other southbound protocols are also available if OpenFlow does not suit a particular purpose. For instance, Cisco's Application Centric Infrastructure (ACI) [9] is another alternative for OpenFlow. However, using such proprietary protocols will hence limit the device's vendor choices.

In order to protect the network from being driven by a malicious controller or to prevent a malicious switch or network device from obtaining any network information, the southbound communication must be operated in a secure channel. OpenFlow suggested securing the southbound communication with Transport Layer Security (TLS).

Projects that do not implement TLS are prone to man-in-the-middle attacks [10]. Attackers will be able to penetrate the OpenFlow networks while remaining undetected. Despite the security risks involved, it is still not implemented in many SDN projects with exceptions such as OpenDayLight [11], HP Virtual Application Network (VAN) SDN [12], and ONOS [13].

One of the main reasons for why TLS is not used by SDN network administrators is that the steps required to configure it correctly can be quite tedious [10]. The TLS implementation requires a Certificate Authority (CA) to generate the CA's key, certificates for the controllers, switches, and then the signing of these certificates with the CA's key. The certificates and devices' keys will then be deployed to the respective devices prior to the actual network deployment. This tedious process is a hindrance for them when adopting the TLS to secure the communication channel.

As illustrated in Figure 1, there are two general types of data planes, the wired and wireless variants. SDN was initially built for the wired data plane and the data center is the main consumer of it. As the SDN technology grows, telecommunication providers became interested in it and started experimenting with it in their network with AT&T supporting both OpenDayLight and ONOS while Verizon, China Unicom, NTT Communication, and SK Telecom are backing ONOS.

However, the original design neglected the wireless data plane that was powered by numerous wireless technologies such as the Fourth-Generation Long Term Evolution (4G-LTE) [14], Universal Mobile Telecommunication System (UMTS) [15], Wireless Fidelity (WiFi) [16], and Worldwide Interoperability for Microwave Access (WiMax) [17]. Hence, it is difficult to manage the heterogeneous wireless data plane [18] even with the current advancement in SDN technology. It is even more complicated when it comes to the security management between these wireless technologies.

Unlike its wired counterpart, the security for the wireless data plane cannot be neglected because anyone within the coverage area of the wireless technology can tap into the network and perform any malicious activities to disrupt the network. Therefore, security becomes a crucial criterion for the wireless data plane before they can deploy it for real usage.

By replacing TLS with Identity-Based Cryptography (IBC) [19], the steps required for system setup will be greatly simplified, improving both the performance availability and bandwidth availability as well as minimizing storage and management of the public keys and therefore saving on costs.

Our Contributions. In this paper, we proposed securing the SDN communication with IBC protocol. To the best of our knowledge, this is the first work that allows multidomain secure communication with IBC in SDN along with its data plane. Our contributions are listed in detail as follows:

(1) We described the security risks involved in SDN and its data plane as well as the reasons it needs to be protected.

(2) We described the reasons the other proposal is insufficient to protect the SDN and why IBC is a preferable method.

(3) We presented the proposal to secure the SDN and its data plane, specifically the wireless data plane, with a multidomain capable IBC protocol.

(4) We provided some application scenarios in which the multidomain IBC protocol can be utilized. We also described how it allows multidomain communication and switch migration in the distributed SDN which previously could not be performed.

(5) We described the application of IBC in helping the communication within the data plane and an analysis to show the possible bandwidth saved with IBC.

2. Background

2.1. Southbound Security. The de facto standard of the SDN southbound protocol, OpenFlow [20, 21], suggested that the southbound communication should be secured with TLS. Projects that do not implement TLS are prone to man-in-the-middle attacks [10]. Adversaries will be able to penetrate the OpenFlow networks while remaining undetected.

In order to prevent the compromise of a controller from the southbound communication channel and a malicious switch or network device from obtaining or modifying any network information, the southbound communication must be operated in a secure channel. Despite this requirement, it is not implemented in many SDN projects because TLS is only an optional feature in OpenFlow specifications and the complicated certificate management. Besides that, the security also comes at a cost to the bandwidth due to the exchange of the certificates for authentication purposes.

2.2. Data Plane Security. The data plane can be further divided into two categories, the wired and wireless data plane. Both data planes may share some security concerns but there are also security concerns that are specific to either one of the data planes. The detailed security concerns will be discussed as follows.

2.2.1. Wired Data Plane. As illustrated in Figure 1, the wired data plane is much simpler compared to the wireless data plane. It involves only the switches, hosts, or any other devices that are connected through the switch. Even if the east/west-bound and southbound communication channels are secure, it does not guarantee that the communication between the devices within the data plane is secure.

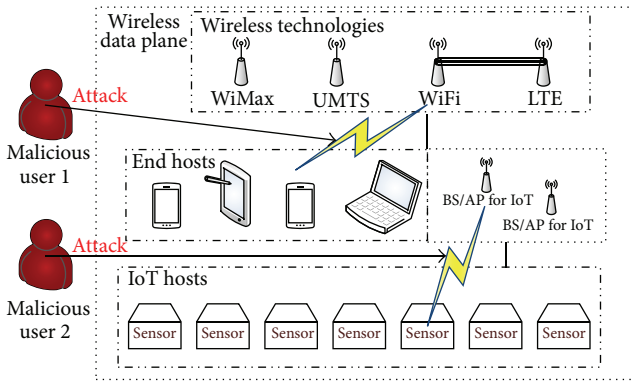

FIGURE 2: Possible attacks within the wireless data plane.

Possible attacks that can be launched within the data plane are Denial-of-Service (DoS) attack and man-in-the-middle attacks that intercept messages from the insecure communication channel and so forth.

2.2.2. Wireless Data Plane. With the growing use of mobile and Internet of Things (IoT) devices, the wireless data plane becomes enormous in size and involves complicated topologies. The capability crisis arises when mobile devices grow at a pace that exceeds the wireless spectrum capability. Therefore, the wireless resource management becomes crucial in order to sustain the network performance for the vast wireless data plane.

This also drives telecommunication providers to look for alternatives to fully utilize their network resources especially for the wireless portion as the wireless bandwidth is limited and expensive. One such solution for them will be to manage their wireless data plane with SDN.

Besides the wireless resource management, certificate management will also be involved if TLS were to be used to secure the data plane. The certificate management can be very complicated due to the enormous amount of devices involved. Besids that, it is also bandwidth consuming to perform the TLS handshake that involves certificate exchanges for authentication.

Figure 2 shows two sample attacks that can happen within the wireless data plane. Unlike the wired data plane, the malicious user does not need to have physical access to the switch to perform any malicious activity. As long as the malicious user is within the wireless coverage range, he/she can simply intercept the wireless communication or even modify the information if the wireless data plane is not protected. This compromises both the data integrity and confidentiality, and hence security is not a luxury feature but a necessity in the wireless communication especially for IoT devices that are deeply involved in personal privacy or even life threatening in the case of IoT devices used in healthcare.

2.3. IoT Application Protocols. In order for IoT devices to be compliant with the one Machine-to-Machine (oneM2M) [22] IoT standard, two widely used application protocols can be used to facilitate the communication within the wireless data

plane: Message Queuing Telemetry Transport (MQTT) [23] and Constrained Application Protocol (CoAP) [24].

2.3.1. MQTT. MQTT is a lightweight asynchronous publish-subscribe messaging protocol that relies on the MQTT broker to facilitate the messages between the publisher and subscriber. MQTT employs Transmission Control Protocol (TCP) to provide a reliable communication channel between the IoT devices. The small header of MQTT protocol (2 bytes only) allows the message delivery with minimal bandwidth and yet reliable connection.

A simple architecture that relies on the MQTT protocol consists of only three main components, broker, publisher, and subscriber. Broker, as the name implies, is a messaging agency or server that distributes the messages between the publisher and subscriber. On the client side, the publisher will send the message to the broker on a particular topic while the subscriber that subscribed to the particular topic will receive the message from the broker. The MQTT protocol also allows multiple subscribers on a topic but multiple publishers on a single topic are not recommended even though it is possible to do so because the subscriber cannot differentiate the source of the message.

2.3.2. CoAP. Similar to MQTT, CoAP [25] is also a widely used lightweight protocol for IoT devices but the similarity ends here. Unlike MQTT, CoAP is based on Representational State Transfer (REST) architecture [26]. Therefore, CoAP relies on the four REST verbs to perform the Create, Read, Update and Delete (CRUD) operations as follows:

(i) POST: create a new resource identifier (ID).

(ii) GET: read/retrieve the information of the resource ID.

(iii) PUT: update the state of the resource ID.

(iv) DELETE: delete a resource ID.

CoAP employs User Datagram Protocol (UDP) to deliver the messages between the IoT devices and uses Uniform Resource Identifier (URI) to address the REST verbs to a particular resource ID. These REST verbs allow it to be integrated with the web, mobile, or even desktop applications easily.

2.4. Revocation. In the Public Key Cryptography (PKC), there are cases that the CA has to revoke the certificates even before they expire. This revocation can happen due to certificate loss by the user, a compromised certificate, an employee that has left the company and hence no longer have the right to use the certificate to access company information, and so forth. Certificate revocation can be done in one of the two common methods [27]:

(1) Certificate Revocation Lists (CRL).

(2) Online Certificate Status Protocol (OCSP).

CRL contains a list of revoked certificates that have yet to expire along with the reasons for revocation. In the case of the long certificate validity, for instance 1 year, this list will be likely very long assuming that the CA has issued a lot of certificates. Users will be required to obtain the complete list from the CA in order to verify the status of the certificate that they are going to use. Hence, this is an inefficient revocation method that involves a lot of network bandwidth to transmit the long list of revoked certificate from the CA.

In OCSP, the user will send a certificate status request to the CA and the CA will check it against its revocation list before informing the user on the certificate's validity. This method reduces the network bandwidth consumption but increases the CA computation cost and the CA will be required to be online to perform verification for the users.

Cooper [27] also worked on the attacks of the revocation list by manipulating the reason codes. If the categorization of the reason codes were not carefully analyzed, the user might be able to abuse it, for example, categorization of the seriousness of the revocation as the key is no longer needed or the key was compromised. The key that is no longer needed might not be handled as quickly as the compromised key and hence this gives the attacker time to use the key as long as it is not updated to the user.

Similar to the PKC, in order to prevent the key misuse of IBC, key revocation has to be done on a compromised node, failed node, and so forth. In the IBC of SDN, key revocation can be performed using one of the two methods:

(i) A trusted third party, mediator.

(ii) A network application on the management plane.

A trusted third party, mediator, was used in PKC [28] and IBC [29] to revoke any malicious user in the system. In this mechanism, the PKG generates a private key of the user and then splits it into two portions, for instance, portion A and portion B. PKG will then send portion A to the user and portion B to the mediator. Similar to the PKG, the mediator is a second trusted party that keeps portion B of the private keys for the users. Besides that, the mediator will keep track of malicious users in the system.

When a user requires his private key, he will send an encrypted message to the mediator for partial decryption. The mediator will then check whether the user is malicious or not. If the user has been compromised, the mediator will deny the operation and the user will be revoked from the system and unable to decrypt any message. If the user is genuine, the mediator will then send the partially decrypted message to the user and the user will then be able to decrypt the partially decrypted message and obtain the plaintext message.

However, the mediator method can be bandwidth consuming since the user and mediator are required to transmit the encrypted message and partially decrypted message through the network. Besides that, the mediator requires extra computing resources in order to perform the partial decryption and has to be online at all times. Figure 3 illustrates the key revocation with the help of the mediator.

The network application on the management plane [30] can be in the form of a firewall, intrusion detection system (IDS) or intrusion prevention system (IPS) application, identity management application, or solely revocation control application. It oversees the network-wide view as per Figure 1

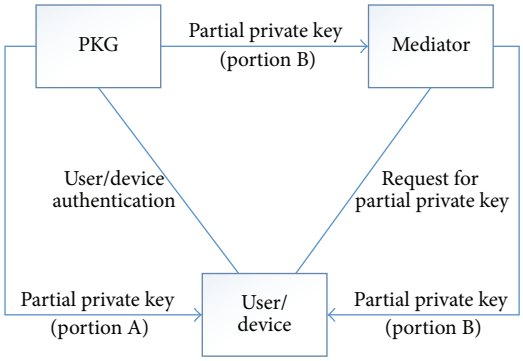

Figure 3: Key revocation with mediator.

(SDN app) and hence will be able to monitor the activities of each node easily.

When the network application detects any malicious behavior in a particular node, it will notify the controller to place the malicious node in a sandbox by blocking all network flows towards the switch. The controller can do so by performing flow removals towards the malicious node. The controller (PKG for the node within its domain) can then analyze the misbehavior. If it is proven safe to continue the communication, the controller can then generate a new private key for the affected switch and distribute this new key to the node.

This network application method might increase the controller's load with the addition of network application and flows removal but a distributed SDN is able to distribute the load with the load balancing mechanism. Hence, this method is preferable in the SDN environment especially where distributed SDN is concerned.

2.5. Identity-Based Cryptography (IBC). IBC was first proposed by Shamir in the form of an identity-based signature scheme [31] in 1985. His idea of IBC was then implemented by Sakai et al. [32] and Boneh and Franklin [33] for the encryption scheme with pairing in the years 2000 and 2001, respectively. Both their implementations went on to form the basis of many IBC researches thereafter, with more being based on the latter.

Similar to the PKC of TLS, IBC requires a Trusted Authority (TA) to act as a Private Key Generator (PKG) that generates keys for the users. In the SDN environment, controllers can also act as PKGs for the switches that are located within its domain.

In PKC, CA is used to generate the public and private key pairs whereas the PKG of IBC generates only the private keys. In IBC, public keys will be derived from the identity of the user; in this case, the user's identity can be in the form of the Media Access Control (MAC) address or any other network identities of the controllers and switches.

With IBC, the users or, in this case, the controllers, switches, or data stores, do not need to store every single public key of every user in the domain or obtain a particular public key from the TA on demand. This in turn saves storage space or network bandwidth that otherwise can decrease the network performance or translate into high system setup costs.

Smart [34], whose research was based on the implementation of Boneh and Franklin, initiated the usage of IBC in key agreement protocol. Chen and Kudla [35] improved Smart's protocol by solving the key escrow problem, allowing for communication between users of multiple TAs and providing forward secrecy.

2.6. Related Research. Santos et al. [36] proposed applying IBC to secure the communications between Master Controller-Secondary Controller (MC-SC), SC-SC, and the client- and server-side or their framework. In their proposal, the IBC protocol was based on the Sakai et al. protocol [32]. However, two issues become apparent if this were to be implemented for practical uses. In their proposal, the Type 1 pairing was used to establish the key. According to the research by Chen et al. [37] and Chatterjee et al. [38], Type 1 pairing is suitable for security levels of up to 80 bits; for security levels higher than 80 bits, the performance will degrade significantly. For details on the 4 pairing types, please refer to [37, 38].

Depending on the usage of the SDN, a security level of 80 bits might be sufficient for a network that manages time-sensitive data or data that might be useless after a short period of time [39]. For a SDN that manages time-insensitive data, a security level of higher than 80 bits should be used. Therefore, the key agreement protocol should be able to support other pairing types.

Another significant disadvantage of their proposal also lies in the key agreement protocol. It does not allow communication between devices that have obtained their private keys from different PKGs. Hence, it becomes impractical especially when it is to be used in the distributed SDN environment as a single PKG might not be sufficient in providing private keys to the entire network. This disadvantage also limits the scalability of the network.

3. SDN Security with IBC

Identity-based key agreement protocol is used to establish the symmetric session key that will secure the SDN communication. Due to the high amount of traffic that will be encrypted, the symmetric key is more preferable to the asymmetric one. The asymmetric keys will be used to derive the symmetric key for session communication.

Our proposal [40] employs the pairing-based key agreement protocol with separate TAs introduced by Chen and Kudla [35] which was originally not meant for SDN. In this implementation, it assumes that the different PKGs share the same domain parameters which is plausible in the case of the SDN setup.

Although it is worth noting that the controller can act as a PKG for all the devices located within the network at the same time, due to the importance of PKG in IBC, it is advisable to have different PKGs generating the private keys for the controllers and switches. By doing so, it can provide better protection to the control plane when the controllers'

A ($S_A = s_1Q_A$, $Q_A = H_1$(A's ID))		**B** ($S_B = s_2Q_B$, $Q_B = H_1$(B's ID))
Choose $a \in_R \mathbb{Z}_q^*$		Choose $b \in_R \mathbb{Z}_q^*$
Compute $T_A = aP$, $W_A = aP_{pub,ctlr2}$		Compute $T_B = bP$, $W_B = bP_{pub,ctlr1}$
	$T_A \longrightarrow$	
	$\longleftarrow T_B$	
$K_{AB} = \hat{e}(S_A, T_B)\hat{e}(Q_B, W_A)$		$K_{BA} = \hat{e}(S_B, T_A)\hat{e}(Q_A, W_B)$
$K_{AB} = \hat{e}(s_1Q_A, bP)\hat{e}(Q_B, as_2P)$		$K_{BA} = \hat{e}(s_2Q_B, aP)\hat{e}(Q_A, bs_1P)$
$K_{AB} = \hat{e}(Q_A, P)^{s_1 b}\hat{e}(Q_B, P)^{s_2 a}$	=	$K_{BA} = \hat{e}(Q_B, P)^{s_2 a}\hat{e}(Q_A, P)^{s_1 b}$

Box 1: Illustration of the establishment of the IBC key agreement protocol.

PKGs are disconnected from the network, thus lowering the risk of the PKGs being exposed or attacked.

However, the controllers' PKGs will be required to reconnect to the network when the private keys expire (can be set at a fixed interval) or when a malicious controller is detected (can be triggered by a network application) whereby a new set of private keys will be needed to secure the control plane.

(1) System Setup. Suppose there are two PKGs, PKG_1 and PKG_2, that generate the private keys for the controllers of the SDN. Each has a public/private key pair, ($P, s_1P \in G_1$, $s_1 \in_R \mathbb{Z}_q^*$) and ($P, s_2P \in G_1, s_2 \in_R \mathbb{Z}_q^*$), respectively, where P and G_1 have been globally agreed on.

Controller A, $controller_A$, is registered under PKG_1 with its private key, $S_A = s_1Q_A$, where $Q_A = H_1$ ($controller_A$'s ID).

Controller B, $controller_B$, is registered under PKG_2 with $S_B = s_2Q_B$, where $Q_B = H_1$ ($controller_B$'s ID). (Note that controllers A and B can also act as PKGs for the switches that are located within their respective domains.) H_1 is a cryptographic hash function; $H_1 : \{0,1\}^* \rightarrow G_1$.

(2) Key Establishment. If controller A wants to communicate with controller B, the IBC key agreement protocol will be initiated to establish the shared session keys. Box 1 illustrates the key establishment of the protocol. Each of controllers A and B picks nonce at random, a and $b \in_R \mathbb{Z}_q^*$, and computes $T_A = aP$, $W_A = aP_{pub,ctlr2}$ and $T_B = bP$, $W_B = bP_{pub,ctlr1}$, respectively, where $P_{pub,ctlr1} = s_1P$ and $P_{pub,ctlr2} = s_2P$. These computed values will then be exchanged between the two controllers.

At the end of the protocol, controller A computes the shared key; $K_{AB} = \hat{e}(S_A, T_B)\hat{e}(Q_B, W_A)$, and controller B computes the shared key; $K_{BA} = \hat{e}(S_B, T_A)\hat{e}(Q_A, W_B)$:

$$\therefore K_{BA} = K_{AB}. \quad (1)$$

Then, the shared session key can be generated by hashing the key; $SK = h_2(K_{AB})$, where h_2 is a secure hash function for the purpose of key derivation. However, this session key does not offer TA forward secrecy and the key escrow issue still exists.

If TA forward secrecy is required and key escrow is not allowed, the previously generated ephemeral keys can also be used to generate a variant of the shared session key; $SK = h_2(K_{AB}, abP)$ as suggested by Chen and Kudla [35]. These shared session keys that have been established by the IBC key agreement protocol will then be used to provide message confidentiality.

3.1. Southbound Security. Controllers that were registered under their respective PKGs can also act as PKGs for the switches located within their domain for southbound communication. Southbound security is more straightforward as it does not usually involve multiple domains. However, during the switch migration from one controller to another, interdomain communication is still required so as to hand over the switch swiftly. Therefore, the same key agreement protocol can also be used for southbound communications.

To apply the IBC key agreement protocol to southbound security, the role of the PKGs will be transferred to the controllers themselves, that is, to generate the private keys for the switches. This reduces the load of the PKGs that manages the control plane and also isolates the two communication channels.

3.2. Data Plane Security

3.2.1. Wired Data Plane. Multidomain key agreement is also helpful in the wired data plane to allow the communication between hosts that obtained their private keys from different controllers through the southbound communication. The key agreement protocol enables the multidomain communication without keeping multiple public keys for each domain. This allows the data plane devices to establish the session by using the identity information and the exchanged parameters.

3.2.2. Wireless Data Plane. The wireless data plane involves multiple wireless technologies and it gets complicated when the end hosts try to establish the communication through heterogeneous backend infrastructure. In order for the heterogeneous communication to take place, a multidomain domain key agreement protocol is used for such a communication.

Despite the multiple wireless technologies used, the underlying protocol to exchange messages may be the same. For example, the two popular communication protocols used by IoT devices, MQTT and CoAP, are able to work with the TCP and UDP, respectively, regardless of the underlying wireless technologies used. Therefore, application of the IBC protocol to either MQTT or CoAP will be able to provide the

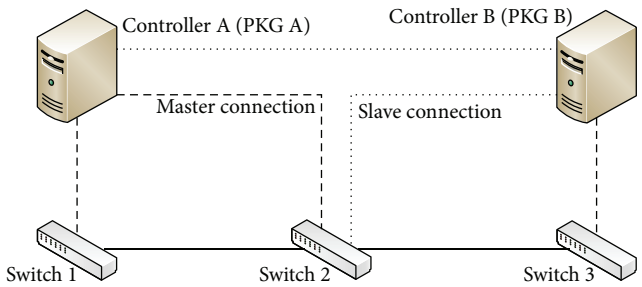

Figure 4: Switch migration for switch 2.

necessary security for the communication between the IoT devices.

4. Application Scenarios

4.1. Southbound Communication. Figure 4 shows the switch migration for switch 2. Switch 2 is allowed to migrate between controllers A and B that also act as the PKGs for switch 2. The load balancing application will notify the respective controller that will be taking over the switch for migration and key establishment will be performed between the switch and controller B.

During this transition period, switch 2 is still able to communicate with switch 1 or switch 3 by using the old key (the key obtained from controller A) and switch over to the new key (the key obtained from the controller that will be taken over) once the key establishment and handover process are completed. When switch migration is completed, the switch can then dispose of the old key and proceed with communication using the new key.

This eases the switch migration process with a single identity information and saves the computing resources at the controller for certificate issuance and management. Besides, it also reduces the bandwidth required for the new certificate distribution in TLS.

4.2. Data Plane Communication

4.2.1. Wired Data Plane. Figure 5 illustrates the interdomain communication within the wired data plane that was made possible with an interdomain key agreement protocol. With the help of the protocol, it allows the communication between switch 1-switch 2 and host A-host B to be established without having a second public key or identity in this case.

Unlike TLS, switch 1 and host A do not need to have controller B issue a new public key to them in order to derive the session key. The same goes to switch 2 and host B without needing a second public key from controller A. Therefore, the key agreement protocol saves computing resources at the controller that generates and manages the certificate. Besides, it also saves the bandwidth that will be used to distribute the certificate.

4.2.2. Wireless Data Plane. The wireless data plane involves heterogeneous wireless technologies and hence the advantage

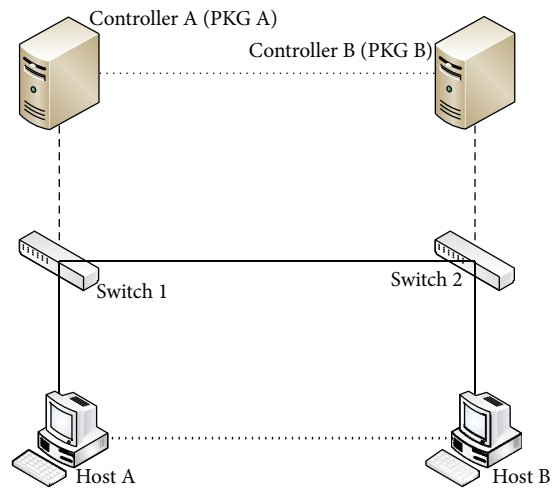

Figure 5: Interdomain data plane communication.

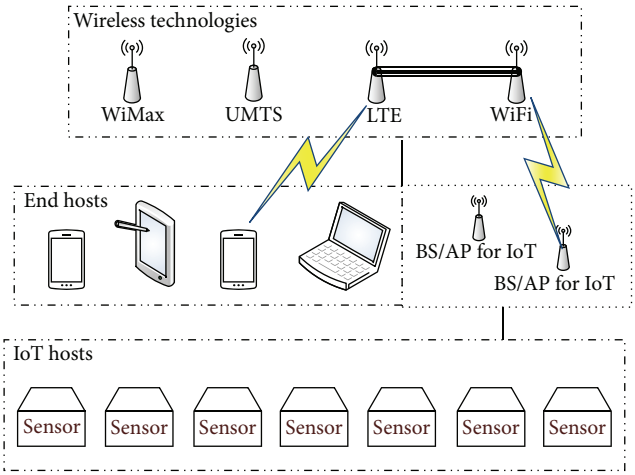

Figure 6: Heterogeneous wireless communication.

of this multidomain capable IBC key agreement protocol actually benefits this data plane communication the most. The certificate management was simplified with the controller managing only the private keys for each of these devices instead of managing multiple certificates for each wireless technology.

Figure 6 illustrates the heterogeneous wireless communication where the mobile device is able to communicate with the base station for IoT devices through the two different wireless technologies. The mobile device connects to the network via Long Term Evolution (LTE) while the base station connects to the network via Wireless Fidelity (WiFi). The communication between the LTE base station and WiFi access point is possible with only the identity information of the mobile devices. It saves the hassle of having multiple certificates for a single device.

Besides that, the IBC key agreement protocol reduces the bandwidth consumption as compared to the TLS for no certificate exchanges will be required. The bandwidth saved

Table 1: Security comparisons between SDN projects.

SDN projects	East/west-bound	Security	
		Southbound	Data plane
OpenDayLight	—	TLS	TLS
HP VAN SDN	—	TLS	TLS
ONOS	TLS	TLS	TLS
D-SDN (Santos et al. [36])	IBC (single domain)	IBC (single domain)	IBC (single domain)
Our proposal	IBC (multidomain)	IBC (multidomain)	IBC (multidomain)

can then accommodate more wireless hosts with the same infrastructure which in turn saves cost.

5. Analysis and Discussions

OpenDayLight and HP VAN SDN implemented TLS to secure the southbound security but neglected the east/west-bound security which is also crucial for a distributed SDN. ONOS secured both the southbound and east/west-bound communications but the system setup is still rather inconvenient since the user is still required to deploy the keys manually.

Santos et al. [36] proposed securing the communications between MC-SC, SC-SC, and the client- and server-side or their framework with IBC. However, there are several flaws in their proposal and their scheme is not suitable for a distributed SDN when interdomain communication is required.

Table 1 is a comparison between the securities of different SDN projects. To simplify the comparison, SDN projects that do not implement a secure channel were excluded from this table. Do note that the protocol can also be added to any open source SDN project as a security module.

Since TCP is the preferred transport protocol to provide reliable communication channel, we chose the MQTT protocol which is one of the most commonly used communication protocols in the world of IoT (relies on TCP) besides CoAP (relies on UDP) and secured it with TLS. Then, we analyzed the communication between the MQTT broker (server) and MQTT client (publisher) to find out the possible bandwidth saved by using IBC instead of TLS.

In the communication analysis, we used a network sniffing tool, Wireshark [41], to monitor the network traffic and the exchanged information between the MQTT client (publisher) and MQTT broker. Figure 7 illustrates a complete TLS handshake for the MQTT protocol and details of handshake messages will be shown in Figure 7.

Figure 8 shows the first message exchange initiated by the MQTT client (in this case, the publisher) to the MQTT broker. The communication started with the TLS handshake mechanism, client hello. In this message, it sends the client's nonce, cipher suites, compression methods, extensions, and any other special features that the client supports to the server or in this case the MQTT broker.

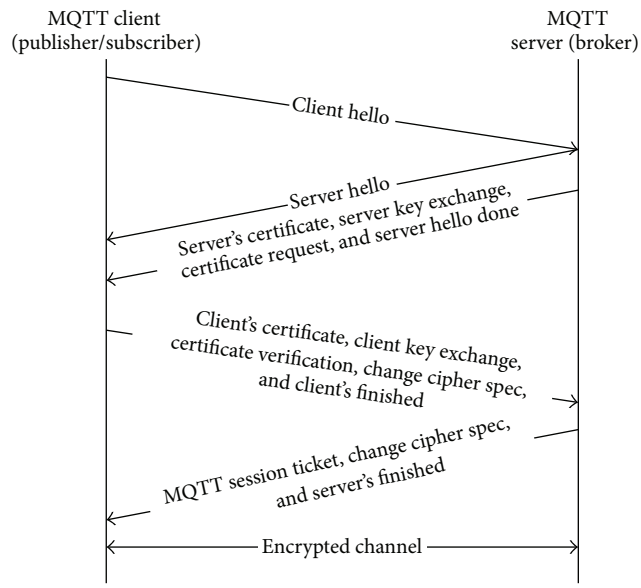

Figure 7: MQTT-TLS handshake mechanism.

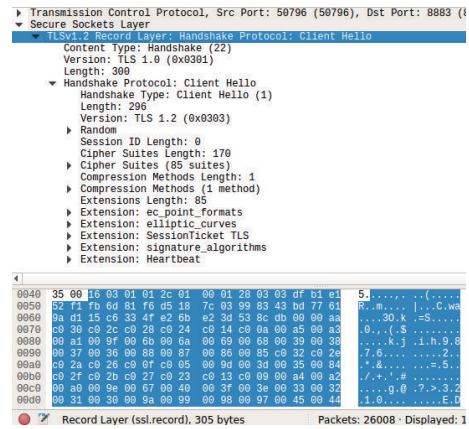

Figure 8: MQTT: client hello message from the MQTT publisher.

Similar to the client hello message, Figure 9 shows the server hello message which is the server's response upon receiving the client hello message. In this message, the server (MQTT broker) will choose the best or most suitable cryptography setting such as the most secure cipher suite supported by the client, compression method used, or any other extensions that were included in the client hello message.

If the IBC protocols were to be applied here, the size of the cipher suites, compression methods, or other features supported might be much lesser initially and hence a smaller client hello message can be used. However, if the IBC protocol is adopted by more organizations, it might grow to a size similar to the TLS protocol. Hence, the bandwidth saved in client hello message can be negligible then. On the other hand, the message length of the server hello message should be similar whether it is in TLS or IBC mode since it only chooses one of the cryptography settings from each type.

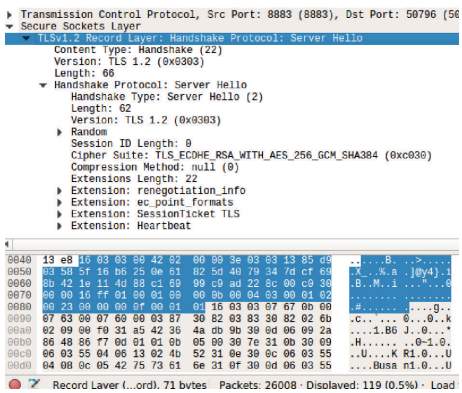

Figure 9: Server hello message from MQTT broker.

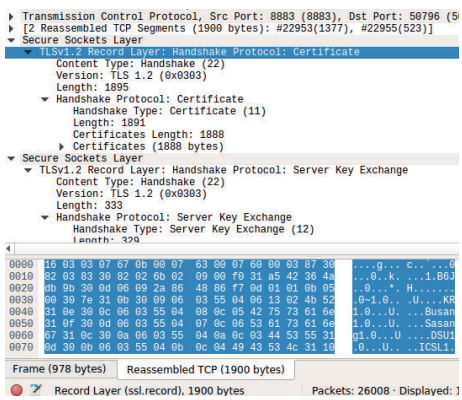

Figure 10: MQTT broker sends its certificate to the client.

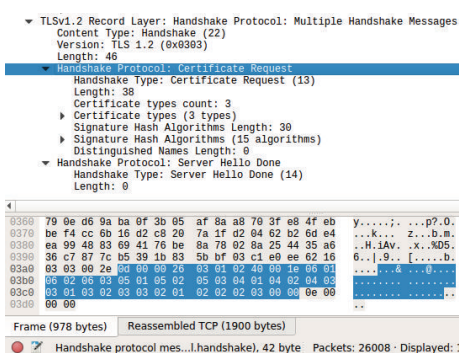

Figure 11: MQTT broker request for client's certificate.

Therefore, the bandwidth saved in the client hello and server hello messages are negligible.

After the MQTT broker (server) sends the server hello message to the client, it will proceed to send its own certificate to the client as shown in Figure 10 and send a certificate request message to the client as shown in Figure 11 so that the server and client can authenticate each other prior to sending any actual data. If IBC protocol is used here, these two messages will no longer be needed because the client will be able to derive the "certificate" from the server's identity and

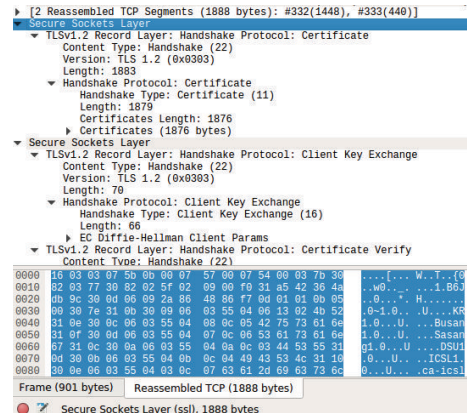

Figure 12: MQTT client sends its certificate to the MQTT broker.

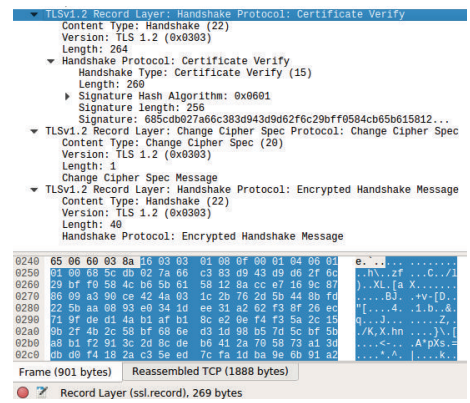

Figure 13: MQTT client sends the certificate verification message to the broker.

the same goes to the server where it will be able to derive the "client's certificate" with the client's identity.

Figure 10 shows that the certificate handshake message used 1900 bytes from the bandwidth while the certificate request message in Figure 11 used 42 bytes of data, resulting in a total of 1942 bytes saved. The bandwidth saved here is significant as the size of the entire client hello and server hello messages is only 376 bytes. By switching it to the IBC protocol, the bandwidth saved here can actually accommodate for another 5 pairs of server-client hello message exchanges.

Figure 12 shows the MQTT client (in this case, publisher) sending its own certificate to the MQTT broker upon receiving the certificate request message from the broker and the information required to verify the client was sent in the certificate verification message as shown in Figure 13 to the broker. Again, if the IBC protocol was to be applied here, these handshake messages will be redundant as the broker is able to derive the "client's certificate" from the client's identity and, with the exchanged nonce and derived "certificates," the broker will be able to verify the client without needing the certificate verification message as well.

Figure 12 shows that the client's certificate used 1888 bytes while the certificate verification message in Figure 13 used 269 bytes of data. Again, a total of 2157 bytes can be saved by switching to the IBC protocol.

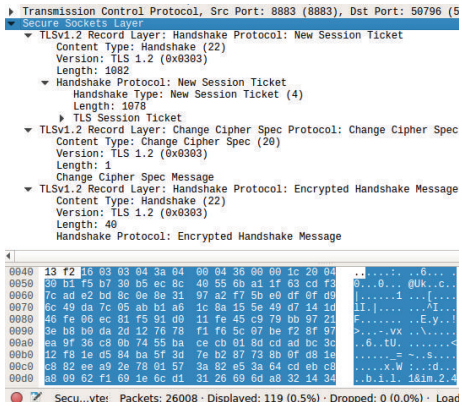

Figure 14: Session ticket from MQTT broker.

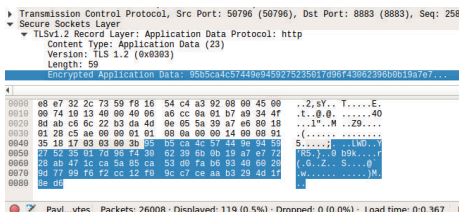

Figure 15: Encrypted channel.

Finally, Figure 14 shows the session establishment of the server-client pair while Figure 15 shows the first application data exchange with the established session's cryptographic parameters. The bandwidth required for these message exchanges should be similar whether it is in TLS or IBC protocol if the same handshake mechanism and symmetric cryptography are used because the size of a session ticket and change cipher spec message is not affected by the cryptography protocol.

The server's finished or the encrypted handshake message shown in Figure 14 and the first encrypted data in Figure 15 will have a similar length regardless of whether TLS or IBC was used because TLS and IBC are used to derive the symmetric key with the provided asymmetric keys via the handshake mechanism and the symmetric key cryptography used to encrypt these messages will be the same for both TLS and IBC. Hence, no bandwidth can be saved here.

By referring to Figure 7, the size of each handshake message is listed as follows:

Client hello message: 305 bytes (refer to Figure 8).

Server hello message: 71 bytes (refer to Figure 9).

Server's certificate: 1900 bytes (refer to Figure 10).

Server key exchange: 338 bytes (5 bytes of header + length, refer to Figure 10).

Certificate request and server hello message done: 51 bytes (refer to Figure 11).

Client's certificate: 1888 bytes (refer to Figure 12).

Client key exchange: 75 bytes (refer to Figure 12).

Certificate verification message: 269 bytes (refer to Figure 13).

Change cipher spec (client): 6 bytes (refer to Figure 13).

Client's finished: 45 bytes (refer to Figure 13).

New session ticket: 1087 bytes (refer to Figure 14).

Change cipher spec (server): 6 bytes (refer to Figure 14).

Server's finished: 45 bytes (refer to Figure 14).

Complete handshake: 6086 bytes.

Based on the analysis of the possible bandwidth that can be saved with IBC, it shows that removing the certificate related messages (server's certificate, certificate request, client's certificate, and certificate verification) from the TLS handshake alone can save up to 4099 bytes per communicating pair. A complete handshake requires 6086 bytes and 4099 bytes are more than half of the bandwidth saved.

Hence, this can lead to lower power consumption as less data are required to be exchanged between the server and client. The bandwidth saved here will also allow the same network infrastructure to accommodate for more communicating pairs and hence improve performance.

Besides that, the multidomain IBC protocol also allows the MQTT client that has the key generated by a controller of a different domain to communicate through the MQTT broker of another domain.

6. Conclusion

There are several notable benefits when securing the SDN communication with IBC; steps required for system setup will be significantly simplified while performance and network bandwidth vastly improved. Furthermore, as a smaller storage space is needed to store the keys, all these will then translate to a decrease in cost.

Besides that, IBC also speeds up the key exchange process since the two communicating parties do not need to obtain each other's public key from the CA to derive the session key. This reduces the time for the key setup and hence less time is spent on the key exchange process.

With this IBC scheme, even the nodes of different subdomains will be able to derive the shared session key. This not only improves the network scalability, but also eases switch migration in the southbound communication and enables interdomain data plane communication.

Lastly, the analysis shows the possible bandwidth that can be saved by switching to IBC; certificate exchanges are the most bandwidth consuming part during a handshake process. By removing the use of certificates in TLS, bandwidth consumption is reduced by as much as 4099 bytes per communicating pair during the handshake process. In other words, more than half of the bandwidth is saved as a complete handshake requires 6086 bytes.

This will lead to lower power consumption of the IoT devices and higher network performance as it can now

accommodate more IoT devices without upgrading the network infrastructure.

Disclosure

Part of this paper was presented in the International Conference on Ubiquitous and Future Networks (ICUFN) 2015.

Competing Interests

The authors declare that they have no competing interests.

Acknowledgments

This research was supported by Basic Science Research Program through National Research Foundation of Korea funded by the Ministry of Education, Science and Technology (Grant no. NRF-2014R1A1A2060021). The third author (Hoon-Jae Lee) was supported by the National Research Foundation of Korea, NRF2011 Project (Grant no. NRF-2011-0023076) and NRF2016 Project (Grant no. NRF-2016R1D1A1B01011908).

References

[1] A. Dixit, F. Hao, S. Mukherjee, T. V. Lakshman, and R. Kompella, "Towards an elastic distributed SDN controller," in *Proceedings of the 2nd ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking (HotSDN '13)*, pp. 7–12, Hong Kong, August 2013.

[2] M. F. Bari, S. R. Chowdhury, R. Ahmed, and R. Boutaba, "PolicyCop: an autonomic QoS policy enforcement framework for software defined networks," in *Proceedings of the Workshop on Software Defined Networks for Future Networks and Services (SDN4FNS '13)*, pp. 1–7, Trento, Italy, November 2013.

[3] R. Skowyra, S. Bahargam, and A. Bestavros, "Software-Defined IDS for securing embedded mobile devices," in *Proceedings of the 2013 IEEE High Performance Extreme Computing Conference (HPEC '13)*, pp. 1–7, Waltham, Mass, USA, September 2013.

[4] R. Braga, E. Mota, and A. Passito, "Lightweight DDoS flooding attack detection using NOX/OpenFlow," in *Proceedings of the 35th Annual IEEE Conference on Local Computer Networks (LCN '10)*, pp. 408–415, Denver, Colo, USA, October 2010.

[5] J. R. Ballad, I. Rae, and A. Akella, "Extensible and scalable network monitoring using openSAFE," in *Proceedings of the Internet Network Management Conference on Research on Enterprise Networking (INM/WREN '10)*, p. 8, 2010.

[6] C. Yu, C. Lumezanu, Y. Zhang, V. Singh, G. Jiang, and H. V. Madhyastha, "FlowSense: monitoring network utilization with zero measurement cost," in *Proceedings of the 14th International Conference on Passive and Active Measurement (PAM '13)*, vol. 7799, pp. 31–41, Springer, Berlin, Germany, 2013.

[7] J. H. Lam, S.-G. Lee, H.-J. Lee, and Y. E. Oktian, "TLS channel implementation for ONOS's east/west-bound communication," in *Electronics, Communications and Networks V*, vol. 382 of *Lecture Notes in Electrical Engineering*, pp. 397–403, Springer, Singapore, 2016.

[8] OpenFlow, https://www.opennetworking.org/sdn-resources/technical-library.

[9] "Cisco Application Centric Infrastructure: Use ACI as a Technology-Based Catalyst for IT Transformation White Paper," http://www.cisco.com/c/en/us/solutions/collateral/data-center-virtualization/application-centric-infrastructure/white-paper-c11-734501.html.

[10] K. Benton, L. J. Camp, and C. Small, "OpenFlow vulnerability assessment," in *Proceedings of the 2nd ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking (HotSDN '13)*, pp. 151–152, Hong Kong, August 2013.

[11] Open Daylight, "Cross Project: Open Daylight Security Analysis," https://wiki.opendaylight.org/view/CrossProject:OpenDaylight_Security_Analysis.

[12] HP, *HP VAN SDN Controller Administrator Guide*, revision 2, 1st edition, 2014, http://h20564.www2.hp.com/hpsc/doc/public/display?docId=c04003114.

[13] ONOS, "Secure OpenFlow connection using TLS/SSL," https://jira.onosproject.org/browse/ONOS-1319.

[14] Y. Zaki, "Long Term Evolution (LTE)," in *Future Mobile Communications: LTE Optimization and Mobile Network Virtualization*, vol. 1 of *Advanced Studies Mobile Research Center Bremen*, pp. 13–33, Springer, Wiesbaden, Germany, 2013.

[15] M. Stasiak, M. Głąbowski, A. Wiśniewski, and P. Zwierzykowski, "Universal mobile telecommunication system," in *Modeling and Dimensioning of Mobile Networks: From GSM to LTE*, John Wiley & Sons, Chichester, UK, 2010.

[16] M. D. Aime, G. Calandriello, and A. Lioy, "Dependability in wireless networks: can we rely on WiFi?" *IEEE Security & Privacy*, vol. 5, no. 1, pp. 23–29, 2007.

[17] S. W. Peters and R. W. Heath Jr., "The future of WiMAX: multihop relaying with IEEE 802.16j," *IEEE Communications Magazine*, vol. 47, no. 1, pp. 104–111, 2009.

[18] F. Bari and V. C. M. Leung, "Automated network selection in a heterogeneous wireless network environment," *IEEE Network*, vol. 21, no. 1, pp. 34–40, 2007.

[19] C. Peng, Q. Zhang, and C. Tang, "Improved TLS handshake protocols using identitybased cryptography," in *Proceedings of the 2009 International Symposium on Information Engineering and Electronic Commerce (IEEC '09)*, pp. 135–139, Ternopil, Ukraine, May 2009.

[20] N. McKeown, T. Anderson, H. Balakrishnan et al., "OpenFlow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[21] S. J. Vaughan-Nichols, "OpenFlow: the next generation of the network?" *IEEE Computer*, vol. 44, no. 8, pp. 13–15, 2011.

[22] Standards for M2M and the Internet of Things: oneM2M Release 1 Specifications, http://www.onem2m.org/technical/published-documents.

[23] D. Locke, "MQ Telemetry Transport (MQTT) V3.1 Protocol Specification," August 2010, http://www.ibm.com/developerworks/webservices/library/ws-mqtt/index.html.

[24] Z. Shelby, K. Hartke, and C. Bormann, "The constrained application protocol (CoAP)," RFC 7252, 2014, https://datatracker.ietf.org/doc/draft-ietf-core-coap/.

[25] C. Bormann, A. P. Castellani, and Z. Shelby, "CoAP: an application protocol for billions of tiny internet nodes," *IEEE Internet Computing*, vol. 16, no. 2, pp. 62–67, 2012.

[26] R. T. Fielding and R. N. Taylor, "Principled design of the modern Web architecture," in *Proceedings of the ACM 22nd International Conference on Software Engineering (ICSE '00)*, pp. 407–416, Limerick, Ireland, June 2000.

[27] D. A. Cooper, "A closer look at revocation and key compromise in public key infrastructures," in *Proceedings of the 21st National Information Systems Security Conference*, pp. 555–565, October 1998.

[28] D. Boneh, X. Ding, G. Tsudik, and C. M. Wong, "A method for fast revocation of public key certificates and security capabilities," in *Proceedings of the 10th Conference on USENIX Security Symposium*, vol. 10, pp. 297–308, Berkeley, Calif, USA, 2001.

[29] X. Ding and G. Tsudik, "Simple identity-based cryptography with mediated RSA," in *Topics in Cryptology—CT-RSA 2003: The Cryptographers' Track at the RSA Conference 2003 San Francisco, CA, USA, April 13–17, 2003 Proceedings*, vol. 2612 of *Lecture Notes in Computer Science*, pp. 193–210, Springer, Berlin, Germany, 2003.

[30] C. Yoon, T. Park, S. Lee, H. Kang, S. Shin, and Z. Zhang, "Enabling security functions with SDN: a feasibility study," *Computer Networks*, vol. 85, pp. 19–35, 2015.

[31] A. Shamir, "Identity-based cryptosystems and signature schemes," in *Advances in Cryptology: Proceedings of the CRYPTO '84*, Section I, pp. 47–53, Springer, 1985.

[32] R. Sakai, K. Ohgishi, and M. Kasahara, "Cryptosystems based on pairing," in *Proceedings of the Symposium on Cryptography and Information Security (SCIS '00)*, Okinawa, Japan, January 2000.

[33] D. Boneh and M. Franklin, "Identity-based encryption from the weil pairing," in *Proceedings of the 21st Annual International Cryptology Conference*, Santa Barbara, Calif, USA, August 2001.

[34] N. P. Smart, "Identity-based authenticated key agreement protocol based on Weil pairing," *Electronics Letters*, vol. 38, no. 13, pp. 630–632, 2002.

[35] L. Chen and C. Kudla, "Identity based authenticated key agreemet protocols from pairings," in *Proceedings of the 16th IEEE Computer Security Foundations Workshop*, vol. 2, pp. 219–233, Pacific Grove, Calif, USA, July 2003.

[36] M. A. S. Santos, B. A. A. Nunes, K. Obraczka, T. Turletti, B. T. De Oliveira, and C. B. Margi, "Decentralizing SDN's control plane," in *Proceedings of the 39th Annual IEEE Conference on Local Computer Networks (LCN '14)*, pp. 402–405, Edmonton, Canada, September 2014.

[37] L. Chen, Z. Cheng, and N. P. Smart, "Identity-based key agreement protocols from pairings," *International Journal of Information Security*, vol. 6, no. 4, pp. 213–241, 2007.

[38] S. Chatterjee, D. Hankerson, and A. Menezes, "On the efficiency and security of pairing-based protocols in the type 1 and type 4 settings," in *Arithmetic of Finite Fields: Third International Workshop, WAIFI 2010, Istanbul, Turkey, June 27–30, 2010. Proceedings*, vol. 6087 of *Lecture Notes in Computer Science*, pp. 114–134, Springer, Berlin, Germany, 2010.

[39] N. P. Smart, V. Rijmen, B. Warinschi et al., "Algorithms, Key Sizes and Parameter Report—2013 recommendations," European Union Agency for Network and Information Security (ENISA), version 1.0, October 2013.

[40] J.-H. Lam, S.-G. Lee, H.-J. Lee, and Y. E. Oktian, "Securing distributed SDN with IBC," in *Proceedings of the 7th International Conference on Ubiquitous and Future Networks (ICUFN '15)*, pp. 921–925, Sapporo, Japan, July 2015.

[41] Wireshark, https://www.wireshark.org/.