Identity-based TLS for Cloud of Chips

Gaurav Sharma, Soultana Ellinidou, Tristan Vanspouwen, Théo Rigas, Jean-Michel Dricot and Olivier Markowitch
Cyber Security Research Center, Université Libre de Bruxelles, Belgium

Keywords: SDN Security, Transport Layer Security, ID-based Cryptography, ID-TLS.

Abstract: In this work, we implement an identity-based Transport Layer Security (ID-TLS) protocol and integrate it on scalable multiprocessor system-on-chip (MPSoC), namely Cloud-of-Chips (CoC), in order to secure the SDN communication on this platform. We select two identity-based encryption schemes that are more likely to meet the performance and resource constraints on the target platform. The schemes are Sakai-Kasahara's identity-based encryption (SK-IBE) and the optimized identity-based encryption (OIBE) for lightweight devices by Guo et al.. The results assert that both the schemes have their computation vs storage trade-off. The SK-IBE algorithm is significantly more computationally efficient than its OIBE counterpart while SK-IBE uses around 30 percent more memory than OIBE. However, the performance results of ID-TLS favor SK-IBE over OIBE. Finally, ID-TLS is integrated in the existing OpenFlow switch and controller implementations. This brings us to a fully functional and secure ID-TLS implementation on CoC, keeping the platform constraints in consideration.

1 INTRODUCTION

The technology landscape of today is dominated by new paradigms, such as internet-of-things (IoT) and internet-of-everything (IoE) which will grow exponentially over the coming years. A wide variety of applications and therefore, extremely versatile hardware solutions are needed to satisfy mobile to high-performance computing requirements. The traditional system-on-chips (SoCs) are no longer capable to support all these applications and hence, a new solution is needed to provide dynamic reconfigurability during run-time and to support a wide variety of use cases. In a recently presented architecture Cloud-of-Chips (Ellinidou et al., 2018; Bousdras et al., 2018), a combination of multiple integrated circuits (ICs) and IC building blocks are interconnected together with different communication speeds and hierarchy levels. The CoC platform contains a printed circuit board (PCB) of multiple ICs where each IC contains scalable processing clusters (PCs). Each PC comprises a combination of high performance and low power cores and thus enable a heterogeneous system architecture. For handling on-chip data communication, network-on-chip (NoC) is installed. The figure 1 below illustrates the CoC architecture. In order to achieve secure communication among PCs and ICs, we propose to implement software defined networking (SDN) paradigm. The traditional routing mechanism employs NoC hardware routers to manage the routes among PCs. However, recent SDN based strategy implements a network manager/controller with global view, which controls the routing in an adaptive manner (Berestizshevsky et al., 2017).

The main idea of the SDN paradigm is to separate the control plane and data plane. OpenFlow (OF) is a common communication protocol used in SDN. OF establishes a unicast communication channel between each individual switch and the controller. It allows the controller to discover OF-compatible switches, creates rules for the switching hardware and also collects statistics. In CoC, the network is separated in two levels: IC and PCB. On the IC level, the hardware routers on network on chip (NoC) are treated as SDN switches and the routing on individual IC is managed by an IC level controller. For the inter IC communication, an SDN switch is placed at the boundary of each IC and is connected to a PCB level controller (Ellinidou et al., 2018). Unfortunately, the SDN paradigm is susceptible to several security breaches (Samociuk, 2015). However, in this paper, we are mainly addressing the communication security among SDN switches (on each IC) and controller. The SDN security requirements are listed in detail in the next section. In addition to the switch-controller unicast communication, multicast communication is

Sharma, G., Ellinidou, S., Vanspouwen, T., Rigas, T., Dricot, J. and Markowitch, O.
Identity-based TLS for Cloud of Chips.
DOI: 10.5220/0007252000440054
In *Proceedings of the 5th International Conference on Information Systems Security and Privacy (ICISSP 2019)*, pages 44-54
ISBN: 978-989-758-359-9

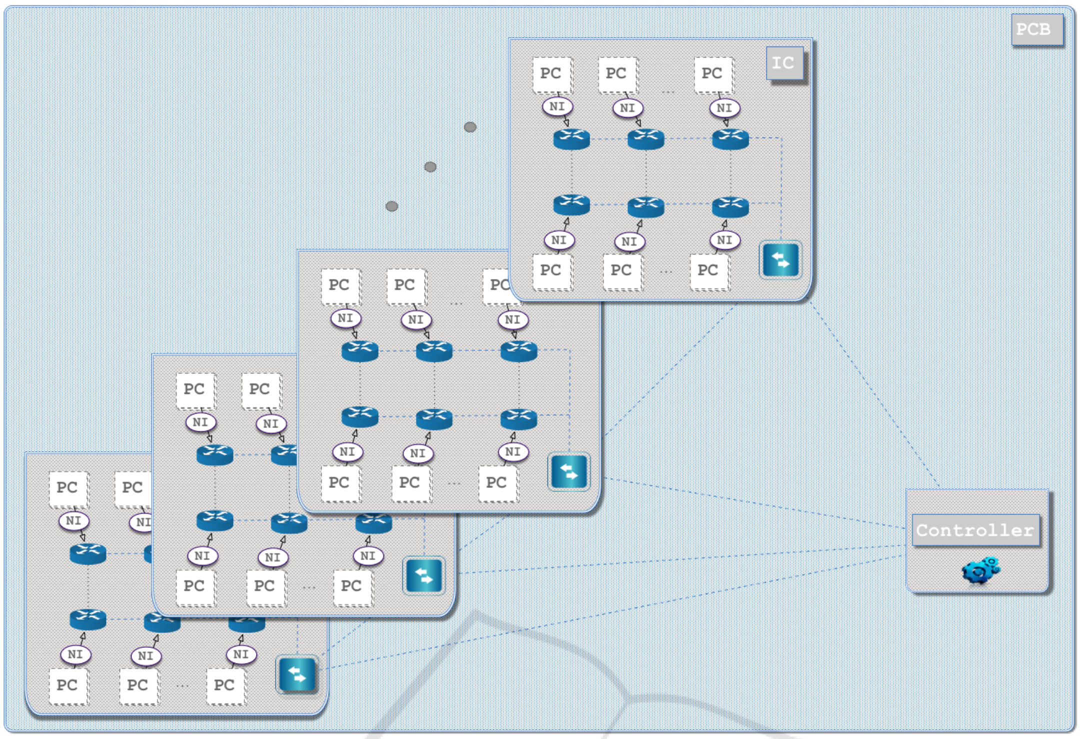

Figure 1: CoC architecture.

also needed to address the issue of secure transfer of routing updates to all/some of the swicthes (Sharma et al., 2018; Sharma et al., 2017).

The existing security solution such as the Transport Layer Security (TLS) protocol, is not well enforced in the current version of the OF standard (Version, 2015). Indeed, the specification reads “the switch initiates a standard TLS or TCP connection to the controller” which means, using TLS is completely optional. Moreover, the public key infrastructure (PKI) overhead includes generation and signing of digital certificates for switches and controller. This makes PKI based solution less acceptable for CoC platform. Therefore, we propose a more suitable solution that fits to unique characteristics of the CoC architecture.

1.1 Our Motivation and Contribution

The deployment of TLS uses complex certificate management which includes certificate validation, certificate lookup and its revocation (Li et al., 2011; Gorantla et al., 2005). A switch would have to store the certificate of controller but also the certificate of other switches (in the case of inter-switch communication), which seems inappropriate from the computation and storage perspective, especially in CoC where these switches are lightweight embedded components on each IC. More precisely, the cost of TLS handshake is our major concern while the TLS Record is very efficient and interesting for our system, since it uses an efficient symmetric encryption (e.g. AES) for communication. In order to use TLS in the CoC context, TLS handshake needs to be modified without compromising the security of authentication and key exchange process. The alternative should not rely on PKI based approach. The main contributions in this research are following:

- We review the IBE literature and select two contrasting approaches well suited to CoC requirements.
- We also propose a practical approach to authenticate the entities participating in the CoC system.
- We implement a modified version of TLS employing two identity-based encryption (IBE) schemes. Furthermore, we compare the performance of ID-TLS to the original TLS protocol using traditional public key schemes and compare the performance in the view of running time and memory consumption.
- Finally, we propose a practical way of integrating this modified version of TLS in existing OF switches and controllers, using “crypto-proxies”.

Rest of the paper is organized as follows: in Section 2, we present TLS Handshake and Record protocol, security requirements for CoC and existing version of modified TLS using IBE. The related work to IBE is presented in Section 3, where it covers most notable IBE contributions. Afterwards the implementation details are presented in Section 4, followed by the results in Section 5. Lastly we summarize our research in the Section 6.

2 BACKGROUND AND SECURITY REQUIREMENTS

In this section, we present TLS protocol, generic modified version of TLS and review security requirements for CoC.

2.1 Transport Layer Security Protocol

The benefits of symmetric and public-key cryptosystems are combined in the Transport Layer Security (TLS) protocol, which can be used to establish an authenticated secure channel between a client and a server. TLS, successor of the well known SSL, has been proposed in the Internet Engineering Task Force (IETF) standard as an improvement of SSL 3.0 in 1999. Over the years, many TLS versions came out, to arrive to the current version of TLS 1.2 in march 2011 (Dierks, 2008). TLS 1.3 is still a draft in 2018, during the writing of this work. TLS is composed of two protocols:

- TLS Handshake Protocol
- TLS Record Protocol

2.1.1 TLS Handshake Protocol

The TLS Handshake protocol's first purpose is to authenticate the server and the client. To that end, PKC schemes such as RSA or DSA can be used. The second purpose is to negotiate different parameters, such as the key to be used for the communication, secured by symmetric encryption. For the key exchange, multiple algorithms are also available such as RSA and Diffie-Hellman. The TLS Handshake is generally initiated by a client request to the server, where the client proposes a list of supported ciphers and hash functions. The server, having received the list, chooses among the options the best cipher and hash function available and sends it back to the client along with its certificate. TLS uses a *X.509* digital certificate which implies the use of CAs and the Public Key Infrastructure (PKI). When a PKC like RSA is used to secure the TLS Handhsake, the procedure continues as follows. After the client verifies the server's certificate, the client and the server exchange R_c and R_s. R_c and R_s are two numbers generated randomly and their purpose is to add protection against replay attacks. The client then chooses a random series of bits, called the premaster secret (PMS). It encrypts the *PMS* using the public key of the server and sends it to the server. The server can decrypt the *PMS* using its private key. Finally, the client and the server, using the *PMS*, R_c and R_s generate the master secret *MS*. This version of the TLS Handshake can be resumed as follows (Roschke et al., 2010):

1. Encryption and hashing functions negotiation between client and server. Generation of random numbers R_c and R_s
2. Transmission and verification of the server certificate including its public key $K_s pub$
3. Exchange of R_c and R_s
4. Client encrypts and transmits *PMS*: $E(PMS, K_s pub)$
5. Generation of the master secret *MS*: $genkey(R_c, R_s, PMS)$

2.1.2 TLS Record Protocol

The TLS Record protocol aims to provide confidentiality through symmetric encryption, using the encryption algorithm negotiated during the TLS Handshake (for example AES, triple DES, etc.) and the master secret *MS* as calculated during the handshake. TLS Record also provides integrity checks of encrypted messages by using hash functions such as HMAC-MD5 or HMAC-SHA.

2.2 Security Requirements

The CoC system is originally comprised of a number of switches and the controller. In order to manage IBE framework for secure communication, the private key generator (PKG, a trusted party) is also a principal component. The overall communication of the system can be separated into two phases. During the first phase, all switches and the controller contact the PKG to get their system parameters and private keys. In the second phase, a switch establishes a secure channel with controller via TLS, to protect the flow of application data, such as OF messages.

2.2.1 Phase 1

Most IBE descriptions specify that the private key must be sent from the PKG to the nodes (the switches and the controller) using a secure channel but they do not specify exactly what this channel could be and its security requirements. The possible threats and solutions are:

- In order to ensure that the only legitimate nodes can receive identity ID and private key, node authentication must be performed by the PKG.
- A counterfeit PKG with different master key generates private keys and IDs for the nodes. This PKG is able to decrypt all the traffic between nodes and controller. In this case, authentication of the PKG by the nodes is also needed.
- An attacker can eavesdrop the response of the PKG and steal the private key of a node. A solution must be there to ensure the confidentiality of communication between a node and the PKG.
- An attacker can sniff the packets exchanged between a node and the PKG and replay them later to obtain a private key.
- An attacker can manage to compromise the integrity of the packets between node and PKG.

2.2.2 Phase 2

This phase refers to switch controller communication where we adopted ID-TLS protocol. The main focus is on the SDN related threats that an ID-TLS could protect against. Following the data flow and interaction among SDN components, Microsoft presents a threat model, STRIDE to meet security requirements CIANNA (confidentiality, integrity, authentication, non-repudiation, availability, authorization). STRIDE stands for Spoofing, Tampering, Repudiation, Information disclosure, Denial of service and Elevation of privileges. In particular, the ID-TLS security solution will address properties such as confidentiality, integrity, authentication and non-repudiation. The other attacks are still applicable and need to counter separately.

2.3 A Modified TLS Protocol Using IBE

A generic modified TLS Protocol for IBE secured communication was introduced by Roschke et al. (Roschke et al., 2010). In the case of a client-server communication, the client derives the public key using server's identity, which can be its *IP* address or its *URL*. Then, the client could directly encrypt the pre-master key (PMS) with the server's public key. The modified TLS handshake steps can be followed from (Roschke et al., 2010). The basic steps are as follows:

1. Exchange of R_c and R_s and algorithm negotiation
2. Client generates the IBE public key of the server $K_s pub$ based on the identity ID_s of the server: $K_s pub = genkey(ID_s)$
3. Encryption of the pre-master key PMS using the public key of the server $K_s pub$: $E(PMS, K_s pub)$ and transmission to the server
4. Both client and server generate the master secret MS using R_c, R_s and PMS: $MS = genkey(R_c, R_s, PMS)$

After the completion of this modified TLS handshake, the client and server can securely communicate with each other using symmetric encryption where master secret is the encryption key.

3 RELATED WORK

The construction of ID-based encryption schemes is generally achieved in one of three ways: prime/composite order pairing or without pairing (refer (Boneh and Franklin, 2001) for standard bilinear pairing definition).

Some notable IBE constructions using prime-order pairing are (Boneh and Franklin, 2001; Sakai and Kasahara, 2003; Boneh and Boyen, 2004; Waters, 2005) and (Gentry, 2006). The first fully functional prime-order pairing based IBE scheme was introduced by Boneh and Franklin (Boneh and Franklin, 2001). In the following text, we will refer this construction as BF-IBE. In 2003, Sakai and Kasahara proposed a new IBE scheme with potentially improved performance (Sakai and Kasahara, 2003). Their scheme SK-IBE is also based on bilinear pairings, however unlike BF-IBE, they use asymmetric pairing and a novel key extraction approach. Boneh and Boyen (Boneh and Boyen, 2004) presented an IBE construction without random oracle. In 2005, Waters (Waters, 2005) presented the first fully secure IBE without random oracle but the scheme is clearly not designed for lightweight devices, as the encryption algorithm requires one multiplication in $\mathbb{G}_T$, three exponentiations (two in $\mathbb{G}_1$ and one in $\mathbb{G}_T$) and an average of $n/2$ (and at most n) multiplications in $\mathbb{G}_1$ where n is the length of the plaintext. Gentry (Gentry, 2006) also proposed a fully secure IBE without random oracle. The authors claimed smaller system parameters and tight security reduction but the key extraction algorithm is quite complex.

Recently, in 2017, Guo et al. (Guo et al., 2017) proposed an optimized IBE (OIBE) especially for lightweight devices. Their encryption algorithm has

Table 1: Performance Comparison of Encryption Schemes.

Schemes	Tool	Required operation(s)				Number of operation(s)		
		$\mathbb{G}_1$	$\mathbb{G}_T$	e	MtP	Hash	Multiplication	Expo.
Boneh-Franklin	Pairing	✓	✓	✓	✓	4	1	1
Sakai-Kasahara		✓	✓	X	X	4	2	1
Boneh-Boyen		✓	✓	X	X	1	2	4
Waters		✓	✓	X	X	1	+- n/2	3
Gentry		✓	✓	X	X	1	5	6
OIBE (*l*=24)		✓	X	X	X	$l \times 2$	$l \times 4$	$l \times 7$
Paterson-Srinivasan	Trapdoor	✓	X	X	✓	2	0	2

three flavors: single/k/multi bit encryption. Their multi-bit encryption scheme is based on the single and k-bit encryption schemes. The idea is to encrypt a L-bit message where $L = l \times k$, which means that if we have a message of 48 bytes (e.g, the pre-master key length in TLS) and $k = 16$ bits then we have to call the k-bit encryption l=24 times. They compared the hardware cost and computational efficiency of their encryption with the Sakai-Kasahara scheme (Sakai and Kasahara, 2003). The OIBE scheme is computationally less efficient than the SK-IBE or other IBE counterparts. Typically, for the L-bit message encryption, the OIBE algorithm is about $\frac{L}{k}$ times slower than other pairing-based IBE schemes. In our system, where the size of the message to be encrypted (the *PMS*) is 48 bytes, the encryption would be 24-times slower.

Another way to construct an IBE is to use a composite-order pairing. Many IBE schemes constructed from composite-order pairings can be found in literature (Waters, 2009; Lewko and Waters, 2010). However, none of them is practical for the CoC platform or in general for lightweight devices.

In 2001, Cocks (Cocks, 2001) proposed a novel IBE scheme based on the quadratic residuosity problem. This scheme is probably the less interesting one due to its inefficiency. Moreover, the scheme is vulnerable to adaptive chosen ciphertext attack. The inefficiency of the scheme comes from the fact that it encrypts messages bit by bit. For example, during the TLS handshake, if we have to encrypt the PMS (384 bits) using a 1024 bit modulus, the sender would have to send $2 \times 384 \times 1024 = 786$ KBytes, which is quite unreasonable for our platform. Another pairing free variant is the construction from trapdoor discrete log groups designed by Paterson and Srinivasan (Paterson and Srinivasan, 2009). The major limitations of this scheme are, the use of a particular curve with a fixed 80-bit security and inefficient private key extraction (Guo et al., 2017).

This scheme is designed for applications where the PKG has a lot of computational power (e.g. military applications) and where the client is very resource-constrained, such as a sensor.
Following the above discussion, two elliptic curve cryptography (ECC) based IBE schemes (Guo et al., 2017; Sakai and Kasahara, 2003) are chosen to modify TLS. Both the schemes have contrasting strong points. It would be interesting to compare them for our platform. The table 1 presents a comparison of all interesting schemes.

3.1 ID-TLS Communication Between Switch and Controller

The switches initialize a TLS connection to communicate securely with the controller. We propose to use a modified version of TLS handshake that utilizes an ID-based encryption scheme to encrypt the pre-master secret (PMS), as shown in figure 2.

4 IMPLEMENTATION DETAILS

In order to implement the two chosen IBE schemes (Guo et al., 2017; Sakai and Kasahara, 2003) and their integration in our modified TLS protocol, the preliminary phase is secure delivery of private key. The next step is to embed the modified IBE protocols to existing TLS framework and then integrate it on CoC platform. The subsections below describe the steps in a more concrete manner.

4.1 Generation of Node Identity and Private Keys

The first step for each node in the system (including the controller) is to contact the PKG, to obtain its identity, its private key and the system parameters. In contrast with common IBE schemes where the nodes choose their identity and provide it to the PKG, in our

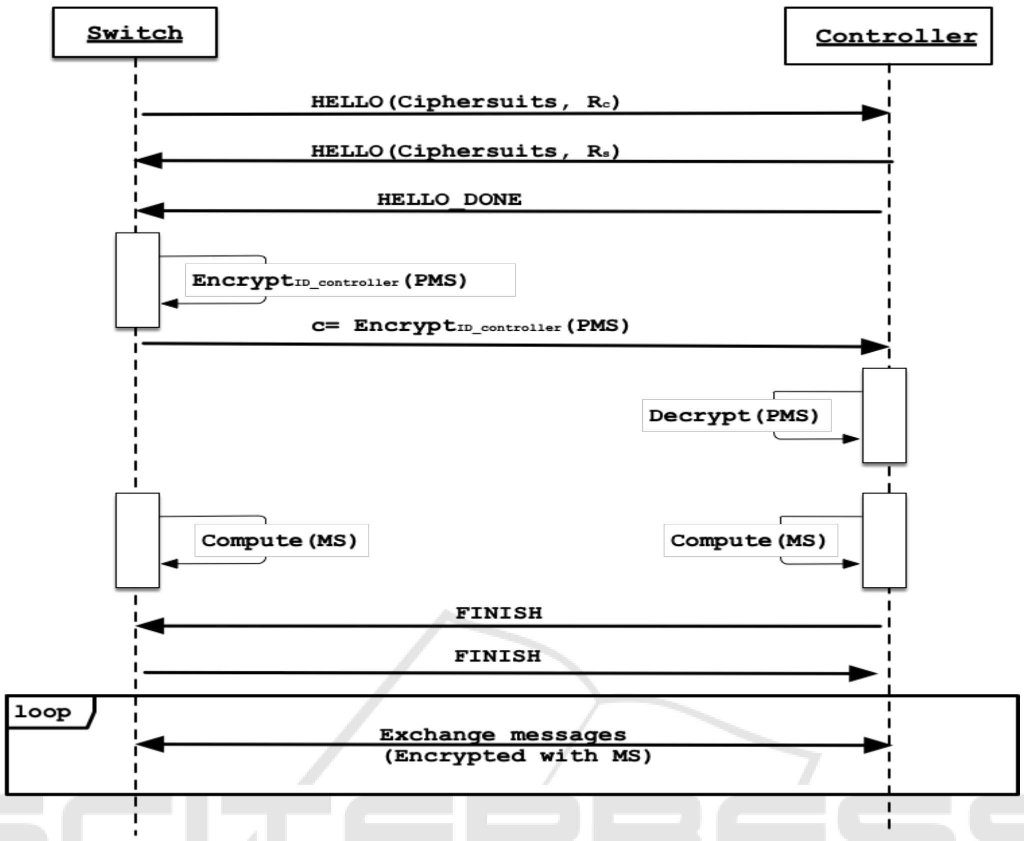

Figure 2: ID-TLS handshake.

implementation the choice of the identity is made by the PKG itself. Unfortunately, implementing this system without any authentication would lead to multiple attacks as explained in section 2.2.1. In order to solve some of the security threats highlighted before, we implemented mutual authentication based on a shared secret symmetric key between the nodes and the PKG. This key is stored at all the ICs (for all switches and controller) at the system integration time. The same key is also used to encrypt the communications and provide confidentiality.

To obtain its identity and private key, the node (switches and controller) encrypts the current timestamp using AES in Galois/Counter Mode (GCM) with the pre-shared key (PSK) and send it to PKG. The same PSK is stored on all ICs at the time of system integration. The private key request to PKG includes the ciphertext, the initialization vector (IV) and the authentication tag. The PKG decrypts the ciphertext using the PSK and matches the authentication tags. Also, the PKG checks whether the decrypted timestamp is between a certain interval (depending on the network, system configuration and speed). Afterwards, the PKG randomly generates a node ID and computes its corresponding private key, following the appropriate IBE scheme. Finally, the PKG encrypts the node ID and private key along with the system parameters using AES-GCM and the PSK. It then sends the ciphertext, IV and tag to the node. The node decrypts the response and stores the ID and private key.

The detailed steps of the modified handshake are as follows:

1. The switch sends a *client hello* request to the controller containing the ciphersuites supported by the client and a random byte string (called R_c). In our system, we propose two ciphersuites making use of ID-based encryption as key-exchange algorithm:

   - **TLS-SK-WITH-AES-256-GCM-SHA384**
   - **TLS-OIBE-WITH-AES-256-GCM-SHA384**

   AES-256 in GCM mode is used for secure communication between switch and the controller, and SHA384 is used as a pseudo random function (PRF) throughout the TLS protocol such as to generate the master secret (*MS*) or during the *FINISH* request to make sure that they both have the same *MS*. SHA384 is also generally used as a parameter of the HMAC function to protect the message integrity which is not needed here because the Galois/Counter Mode already provides message integrity protection.

2. The controller responds by sending the ciphersuite, a random byte string (called R_s) and other relevant information such as session ID
3. The controller notifies the client that the hello procedure is done
4. The switch generates the PMS which is a random 48-byte string, encrypts with the ID of the controller.
5. This step is common for the controller and switch, as both have to compute the *master secret* (*MS*) using the PMS, R_c and R_s:

   $$MS = PRF(PMS, R_c, R_s)$$

   where *PRF* is a pseudo random function.
6. Finally, the controller and switch send the *FINISH* request which is the first protected packet with the newly generated master secret. They verify that the content of the message is correct: $Verif(PRF(MS, \text{“}finished\text{”}, Hash(handshake - message))) == 1$
   The handshake-message is the concatenation of all the handshake communications such as client hello, server hello, server hello done and key exchange messages.
7. When both parties have sent, received and validated the *FINISH* requests of each other, they can start sending encrypted application data using AES-256 in GCM mode.

In order to ensure that only nodes authenticated by the PKG can establish a secure connection with the controller, we use token-based authentication. The steps are as follows:

1. The PKG and the controller have a pre-shared secret key K_PKG_C
2. During the key extraction process, the PKG generates an $authToken = HMAC_{K\_PKG\_C}(nodeID)$ and delivers to all switches. The authToken is transmitted encrypted along with the private key
3. During the TLS handshake, the node sends the encrypted PMS, his ID and a token = authToken $\oplus$ PMS[2:len(authToken)+2]. Note that the first two bytes of the PMS are the protocol version, so they are deterministic and cannot be used to encrypt the token. The controller decrypts the PMS, XORs it again with the token to obtain the authToken and then checks its validity using the node ID and the K_PKG_C

4.2 ID-TLS on CoC

In order to realize the key extraction and ID-TLS, as a proposed solution to CoC, precisely the steps are detailed below.

1. The switch and controller encrypt a timestamp along with PSK and send the resulting ciphertext, IV and tag to the PKG
2. The PKG decrypts the ciphertext, checks the tag and timestamp and then generates the node ID. Now, PKG runs the key extract algorithm of SK or OIBE, to get the private key corresponding to the newly generated ID. Finally, PKG generates the authToken for each switch ID.
3. The PKG sends encrypted the generated ID, private key and authToken along with the parameters of the SK or OIBE scheme. Depending on the parameters of each IBE scheme, the content of the encrypted packet can be as follows (For more details about parameters, refer (Guo et al., 2017; Sakai and Kasahara, 2003)):
   - SK: $(ID, authToken, d_A, P_{pub}, P_1, P_2, g = \hat{e}(P_1, P_2))$
   - OIBE: $(ID, authToken, d_{ID}(d_1, d_2), (u_1, u_2, h_1, h_2, g_1, g_2))$
4. The switch decrypts and verify the received message. During the TLS handshake, the switch encrypts the PMS using chosen IBE scheme and sends the encrypted PMS along with its identity and a token, proving its membership of the system
5. After the TLS handshake and the generation of the master secret (MS), the switch and the controller can communicate securely.

5 RESULTS

In order to analyze the running time and the memory consumption of the ID-TLS handshake, we compare it with TLS handshake using **RSA**. We will first analyze the performance of ID-based encryption and decryption compared to **RSA** in isolation and then we will look at the general performance of the TLS handshake. We implemented our solution integrating **mbed TLS** (also known as **PolarSSL** in the past) with **PBC** library. For the bilinear pairing map, we choose type D pairing which implements **MNT** curves with embedded degree 6. The order of the curve is around 170-bits (base field) and therefore the security will be 1024-bit in the subgroup of F_{p^6}, which is considered

Table 2: Running time comparison of encryption/decryption and key extract functions of the SK, OIBE with an average of 100 repetitions.

Scheme	Encryption time (ms)		Decryption time (ms)		Extract time (ms)	
	CPU time	Real time	CPU time	Real time	CPU time	Real time
SK	2.639	2.643	5.525	9.967	5.314	10.714
OIBE	57.610	136.308	264.445	443.481	11.069	21.456
RSA	0.193	0.156	3.257	3.331	-	-

Table 3: Speed performance of the TLS handshake and private key extraction on an average of 100 repetitions.

Scheme	Handshake client		Handshake server		Get private key	
	CPU time	Real time	CPU time	Real time	CPU time	Real time
SK	2.989	93.869	6.099	59.764	30.037	78.973
OIBE	57.884	582.496	266.348	583.022	30.093	87.707
RSA	0.674	89.216	3.633	44.534	-	-

good enough (Lynn, 2010). We compare the following three ciphersuites:

- **TLS-SK-WITH-AES-256-GCM-SHA384**
- **TLS-OIBE-WITH-AES-256-GCM-SHA384**
- **TLS-RSA-WITH-AES-256-GCM-SHA384**

5.1 Performance Results Without TLS Handshake

As expected, the OIBE *encryption* and *decryption* algorithms are far slower than both RSA and SK. Our experimental results show that the *encryption* is 22-times slower in CPU time and 52-times slower in real time while *decryption* is 48-times slower in CPU time and 45-times slower in real time. However, in comparison with the RSA scheme, the performance of the SK scheme is respectable. Also, note that the code of the RSA scheme is fully optimized by the authors of the mbed TLS library while our implementation of SK is not optimized. Therefore, with a more optimized SK implementation we could be closer to the performance of the RSA scheme. Furthermore, the *key extract* is also two times faster in SK as compared to OIBE. The following table 2 describes the comparative results of their timings.

5.2 Performance Results of the ID-TLS Handshake

The running time for private key extraction and ID-TLS handshake is presented in table 3.

Note that for SK and OIBE, the TLS handshake also comprises the token management used for authentication of the switches to the controller. The RSA handshake does not include this mechanism. From table 3, we see that the CPU time of the handshake for all three schemes is almost the same as the CPU time of the encryption and decryption as shown in table 2 above. Therefore, the cost of token-based authentication mechanism seems negligible and does not add more computational overhead. Furthermore, we claim that the RSA scheme seems faster than SK but the difference is trivial. In the case of a real-time application, the RSA handshake would have to include the verification of server certificate and in that case, SK could overtake.

The real time cost of the client and server handshake is very high (around 0.5 second) for OIBE, which is due to the slow encryption and decryption algorithm. The time of the "Get private key" column represents the interval from the moment when a node sends an encrypted private key request to the PKG until the moment it receives its parameters (private key, ID etc.) and initializes the elements it needs to use IBE in TLS. The speed of the two IBE is about same, SK being slightly faster.

5.3 Memory Usage

In comparison with the SK encryption, OIBE uses far less memory with a memory consumption around 12 Kbytes. In order to estimate the memory usage for a TLS secure communication, we transfer a webpage between a client and a server. The client used approximatively the same memory, irrespective of the encryption used in the TLS handshake. However, on the server side, OIBE uses significantly more heap memory because it has to store the big hashmap used for the decryption. The following table 4 represents the peak heap memory consumption of our testing application. The OIBE scheme uses less heap memory during the encryption than SK, but the memory usage of an actual application is comparable to that of SK. This is because OIBE has to store more cryptographic parameters for its operation, so the initialization of these parameters consumes additional memory. In addition, the memory usage of the server for the OIBE scheme is much higher than that of the SK scheme. Finally, the OIBE scheme at server requires around 7 seconds to perform the initialization of the system before clients can contact it, while the SK initialization is much faster. So, in applications where speed is critical, the SK scheme seems more suitable for an ID-based version of TLS, while in applications where lower heap memory usage can make a difference, the OIBE scheme might offer a viable alternative.

Table 4: Memory usage.

Scheme	Memory usage client	Memory usage server
SK	3148 KB	3296 KB
OIBE	3160 KB	22752 KB
RSA	3344 KB	3332 KB

5.4 Results on CoC

In order to investigate the integration of above modified TLS, we need SDN supporting switches and controller. The OpenVSwitch (OVS) and Ryu controller are chosen to run on an SDN emulator Mininet (Mininet, 2014). OVS is written in C, while Ryu is mainly written in Python. Rather than following direct integration of the secure channel in OVS and Ryu, we opted for a solution that allows using them more like "black-boxes", with minimal or no modifications to their pre-existing functionality. We call them "crypto-proxies". These proxies handle all the tasks detailed above: obtaining the private key, establishing a TLS connection and then handling the encryption and decryption of messages accordingly.

On the CoC platform, it is easy to realize these proxies in the network interface (NI) of the IP core. Moreover, there are several advantages of the proxy approach as follows:

- Our solution is independent of the chosen OF switch or controller
- New security solution is easy to adopt as simple as replacing the current proxy with a new one
- This approach bypasses the communication channel originally used between the switch and the controller (TCP), which means that the data can now be transferred over UDP or by any other means, eventually a new protocol

We claim that there is no security compromise while using proxies. The message generated at IP core is encrypted and decrypted in its own NI and to the best of our knowledge, there is no existing breach in this situation. Using the setup described above, we measured the flow establishment time (defined below) for the default insecure version of OF as a baseline. We then measured the secure channel setup and flow establishment time for the OF secured with the ID-TLS proxies.

In the TLS case, secure channel setup refers to the execution time of the TLS handshake (for one switch). To measure the flow establishment time, we used the `pingall` command, which makes each host ping all the other hosts in the network. In a fresh run of the network emulation, the first time `pingall` is executed, the switches have to contact the controller to create their flow tables. For all subsequent executions, the relevant flow entries are already inside the switch's flow tables and no further communication with the controller is required. We define the flow establishment time as the average execution time of the first `pingall` command minus the average execution time of subsequent `pingall` commands (which is $35.51ms$ in our test setup). Our results are displayed in Figure 4 below:

The Sakai-based variant of TLS clocks in at about $100ms$. The OIBE version of TLS is about 6 times slower. It is important to note that in different scenarios, the TLS cost will remain constant irregardless of the number of switches. However, we calculated that the average AES-GCM encryption-decryption times in our testing environment is $0.0045ms$ (Averaged over 1000 executions on actual OF messages, Standard deviation: $0.001ms$).

6 CONCLUSION

This paper demonstrated that ID-TLS can be employed to secure the SDN communication on CoC. We analyzed the performance of ID-TLS with traditional PKI based version of TLS in the view of computation and storage efficiency. In particular, our basis for comparison is to analyze the performance of traditional protocols and their TLS versions. The SK and RSA protocols have comparable performance while OIBE is significantly slow. However, the OIBE protocol while slower during the encryption and decryption, has significantly lower heap memory consumption during the encryption. In a client application where speed is critical, the SK scheme seems more suitable while in applications where lower heap memory usage can make a difference, the OIBE

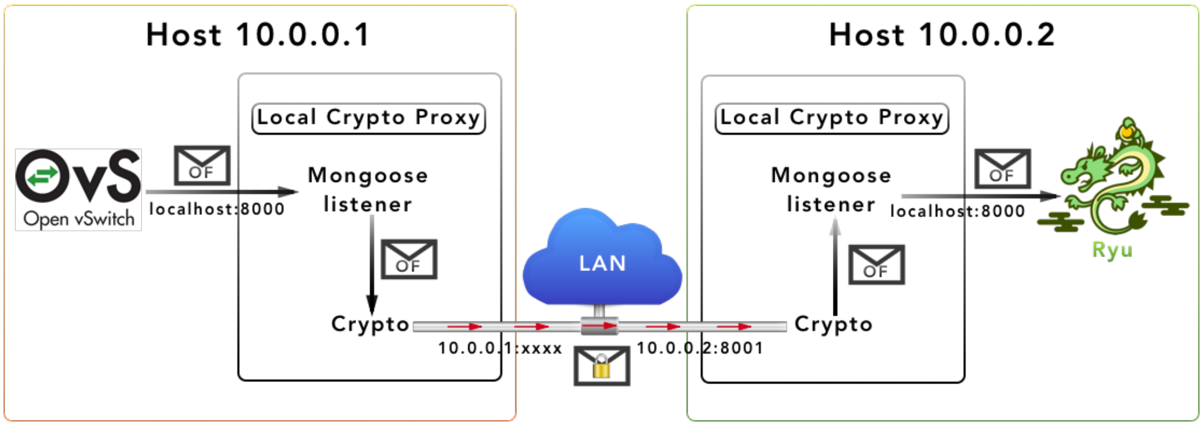

Figure 3: "crypto proxy" architecture.

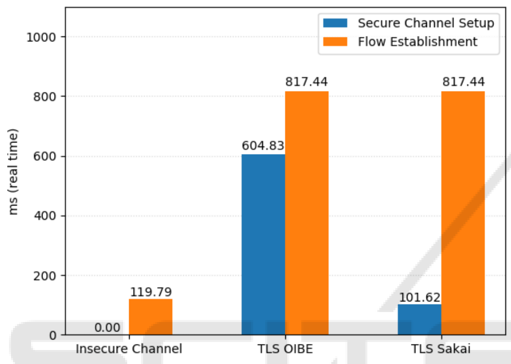

Figure 4: Secure Channel setup and Flow Establishment times, averaged over 10 executions.

scheme might offer a viable alternative. The integration of ID-TLS in existing OF nodes makes the solution more appealing, especially for embedded systems. Moreover, the proposed solution is not limited to CoC but applicable to all such SDN instances where certificate management is complex. The realization of proposed solution on CoC hardware is interesting for further exploration.

REFERENCES

Berestizshevsky, K., Even, G., Fais, Y., and Ostrometzky, J. (2017). Sdnoc: Software defined network on a chip. *Microprocessors and Microsystems*, 50:138–153.

Boneh, D. and Boyen, X. (2004). Efficient selective-id secure identity-based encryption without random oracles. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 223–238. Springer.

Boneh, D. and Franklin, M. (2001). Identity-based encryption from the weil pairing. In *Annual international cryptology conference*, pages 213–229. Springer.

Bousdras, G., Quitin, F., and Milojevic, D. (2018). Template architectures for highly scalable, many-core heterogeneous soc: Could-of-chips. In *2018 13th International Symposium on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC)*, pages 1–7. IEEE.

Cocks, C. (2001). An identity based encryption scheme based on quadratic residues. In *IMA International Conference on Cryptography and Coding*, pages 360–363. Springer.

Dierks, T. (2008). The transport layer security (tls) protocol version 1.2.

Ellinidou, S., Sharma, G., Dricot, J.-M., and Markowitch, O. (2018). A sdn solution for system-on-chip world. In *Software Defined Systems (SDS), 2018 5th Int. Conf. on*, pages 14–19. IEEE.

Gentry, C. (2006). Practical identity-based encryption without random oracles. In *Annual Int. Conf. on the Theory and Applications of Cryptographic Techniques*, pages 445–464. Springer.

Gorantla, M. C., Gangishetti, R., and Saxena, A. (2005). A survey on id-based cryptographic primitives. *IACR Cryptology ePrint Archive*, 2005:94.

Guo, F., Mu, Y., Susilo, W., Hsing, H., Wong, D. S., and Varadharajan, V. (2017). Optimized identity-based encryption from bilinear pairing for lightweight devices. *IEEE Trans. Dependable Secur. Comput.*, 14(2):211–220.

Lewko, A. and Waters, B. (2010). New techniques for dual system encryption and fully secure hibe with short ciphertexts. In *Theory of Cryptography Conference*, pages 455–479. Springer.

Li, H., Dai, Y., and Yang, B. (2011). Identity-based cryptography for cloud security. *IACR Cryptology ePrint Archive*, 2011:169.

Lynn, B. (2010). The pairing-based cryptography (pbc) library. Accessed: 2018-04-25.

Mininet (2014). Mininet project. http://mininet.org/. Accessed: 2018-04-25.

Paterson, K. G. and Srinivasan, S. (2009). On the relations between non-interactive key distribution, identity-based encryption and trapdoor discrete log groups. *Designs, Codes and Cryptography*, 52(2):219–241.

Roschke, S., Ibraimi, L., Cheng, F., and Meinel, C. (2010). Secure communication using identity based encryption. In *IFIP International Conference on Communications and Multimedia Security*, pages 256–267. Springer.

Sakai, R. and Kasahara, M. (2003). Id based cryptosystems with pairing on elliptic curve. *IACR Cryptology ePrint Archive*, 2003:54.

Samociuk, D. (2015). Secure communication between openflow switches and controllers. *AFIN 2015*, 39.

Sharma, G., Kuchta, V., Sahu, R. A., Ellinidou, S., Markowitch, O., and Dricot, J.-M. (2018). A twofold group key agreement protocol for noc based mpsocs. In *2018 16th Annual Conference on Privacy, Security and Trust (PST)*, pages 1–2. IEEE.

Sharma, G., Sahu, R. A., Kuchta, V., Markowitch, O., and Bala, S. (2017). Authenticated group key agreement protocol without pairing. In *International Conference on Information and Communications Security*, pages 606–618. Springer.

Version, O. S. S. (2015). 1.5. 0 (protocol version 0x06). *Open Networking Foundation*.

Waters, B. (2005). Efficient identity-based encryption without random oracles. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 114–127. Springer.

Waters, B. (2009). Dual system encryption: Realizing fully secure ibe and hibe under simple assumptions. In *Advances in Cryptology-CRYPTO 2009*, pages 619–636. Springer.