DOI 10.12694/scpe.v20i1.1473
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 20, Number 1, pp. 181–189. http://www.scpe.org © 2019 SCPE

AN EFFICIENT ZERO-KNOWLEDGE PROOF BASED IDENTIFICATION SCHEME FOR SECURING SOFTWARE DEFINED NETWORK

HAMZA MUTAHER[*] AND PRADEEP KUMAR [†]

Abstract. Researchers and enterprise networks are extensively adopting Software Defined Networking (SDN) due to its feature of decoupling data and control planes from network devices which enable them to implement new networking ideas to solve networking issues like the lack of security. Communication between data and control planes in SDN faces various security issues where many users in data plane approach controller device in control plane to gain networking policies. In this paper, we proposed an efficient Zero-knowledge proof based identification scheme for securing the SDN controller during data and control plane communication. This scheme ensures that only users who prove their knowledge about secrecy without revealing the actual secret or any other information about it can communicate with the controller. The computation cost, communication cost and storage overhead analysis are discussed along with the security analysis to validate the efficiency of the proposed work.

Key words: SDN, Controller, Identification, Zero Knowledge Proof, data plane, control plane.

AMS subject classifications. 68M12

1. Introduction.

Software Defined Networking (SDN) is one of the most important networking topics discussed in recent years [1]. SDN is defined in the article of Open Network Foundation (ONF) [2] as: In the SDN architecture, the control and data planes are decoupled, Network intelligence and state are logically centralized, and the underlying network infrastructure is abstracted from the applications. The Idea of SDN is to separate the control plane from the data plane in network devices. Control plane manages the network policies and has a wide view of the whole network, where the data plane follows the rules and policies sat up by control plane such as packet forwarding policies [3]. The architecture of SDN contents of 3 layers, Application layer, Control layer and the Infrastructure layer. The application layer contains the network applications like network management applications and business applications, and it communicates with the control layer through northbound APIs. Control layer contains the brain of the network (the controller) which has the complete ability of network management and runs the Network Operating System (NOS). Infrastructure layer contains the switching devices. Switching devices are responsible to forward the network packets from Source to destination as per controller decisions and communicate with control layer through southbound APIs. The architecture of SDN illustrated in Figure 1.1. SDN came in to picture to overcome the complexity of the traditional network and to add some new features which facilitate network innovations. These features are explained in Table 1.1 and as follows:

i. Centralization: It is one of the SDN features that ease the deployment of new policies and applications among network devices. It also mitigates the complexity of network management which is not there in a traditional network, i.e. No central point of management. In SDN, the controller is the networking device runs NOS. NOS is responsible for managing and controlling the centralized devices like controller. The controller is responsible for managing the SDN network and has the full knowledge about every device in the network as it is a centralized management point [4].
ii. Scalability: It is an SDN feature that allows the network to have a massive number of devices and enable the network to add a new number of devices into the network as per network needs. Distributed controllers are used to scale up the network and make the network highly available [5].
iii. Heterogeneity: SDN network allows a different type of networks to communicate with each other as per controller decisions. The controller is responsible for this communication as it can manage heterogeneous networks working with different routing protocols. Applications and devices from different platforms also can communicate and exchange services among each other in an SDN network [6].
iv. Programmability: SDN simplifies the network management and innovation by the feature of programmability. Programmability allows the controller to program new rules and deploy them into data plane devices in order of network stability. The third parties can simply program and execute their business applications

[*]Department of Computer Science and IT, Maulana Azad National Urdu University, India (hamzamutaher@gmail.com)
[†]Department of Computer Science and IT, Maulana Azad National Urdu University, India (drpkumar1402@gmail.com)

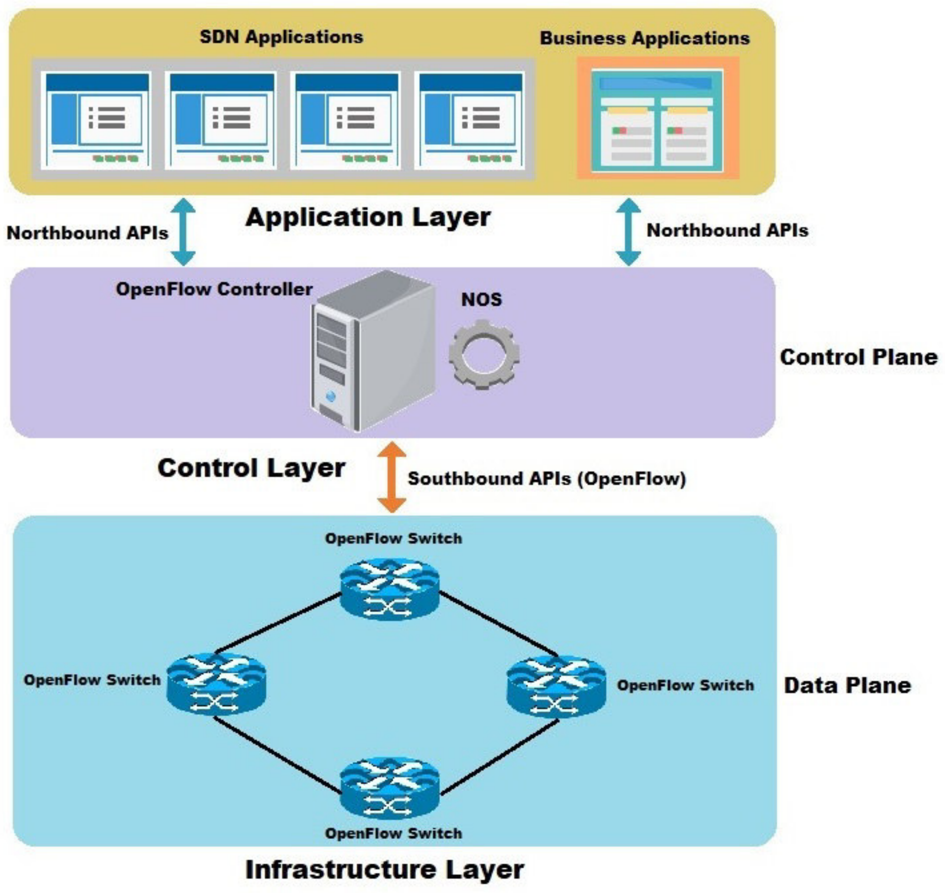

Fig. 1.1. *SDN Architecture*

Table 1.1
Difference between SDN and traditional network

Feature	SDN Network	Traditional Network
Centralization	Yes	No
Scalability	Yes	No
Heterogeneity	Yes	No
Programmability	Yes	No

in SDN environment. Researchers and network developers use the programmability feature to implement their new ideas and innovations [7, 8].

OpenFlow is a standard networking protocol used to communicate between control and data planes through southbound APIs in SDN. OpenFlow standard has various versions like 1.0.0 [9], 1.1.0 [10], 1.2.0 [11], 1.3.0 [12], 1.4.0 [13] and 1.5.0 [14]. OpenFlow-Configuration (OF-CONFIG) protocol is proposed by ONF [15] as a configuration protocol. The controller device which works with the OpenFlow protocol called OpenFlow controller. OpenFlow controller can add or remove entries in the flow table of the OpenFlow switch. The switch which works with OpenFlow protocol is called OpenFlow switch. OpenFlow switch contains two parts described in [16] as follows:

i. A flow table: Every OpenFlow switch has a flow table. Each entry of flow table is associated with an action decided by the controller.

ii. A secure channel: It is a communication channel that connects between OpenFlow switch and controller. It allows commands and packets pass through it using OpenFlow protocol.

The communication between control plane device (controller) and data plane devices (switches) must be secured as many commands and packets are sent between them. The importance of security in this case is to ensure the integrity of these commends and packets from being modified by third party. These commands and packets contain sensitive rules and policies of the network generated by the controller to be placed in switch's flow table entries. If these commands and packets manipulated, many changes may occur to the network and may the controller will not have the full control on switches. Hackers may also change these commands and packets to gain access to the controller. In this case, the hackers can control the whole network and disable some services. In the rest of this paper, we provide the related work in Section 2. The background and motivation are explained in Section 3. Section 4 talks about the proposed scheme along with its phases. We discussed the result of the proposed scheme in Section 5. Then we conclude this paper in the last section.

2. Related Work. Security between data and control planes in SDN is one of the major aspects that researchers and network administrators give much effort to ensure its efficiency. This effort aims to make robust authentication between network users in data plane and controller device in the control plane in order to manage the performance of the network and allow different applications take place in SDN environment securely. Many of researches have been conducted regarding this issue and they are as follow:

Osamah et al. [17] investigated the security threats occur on the SDN control channel by the Denial of service attack (DoS), then they evaluated the impact DoS attack by simulating a DoS attack against the controller device. They used the controller benchmarking (cbench) [18] as a tool to calculate and evaluate the performance of the SDN controller during normal operation (NO). After investigation and evaluation, they proposed a lightweight information hiding mechanism to avoid threats on SDN control channel. This authentication mechanism obscures the sensitive information from being revealed using information hiding algorithm, unlike the cryptography mechanisms which mingle the sensitive data. Finally, they suggested various recommendation to enhance the security of the proposed mechanism.

Mengmeng et al.[19] aimed to develop a security solution technique to guarantee the integrity of SDN flow rules. The authors analyzed the previous techniques which protect and control the SDN flow rules and then designed a PERM-GUARD. PERM-GUARD is a scheme for managing permissions and authenticating the flow rules in the SDN network. This scheme adds some components and functions to SDN architecture as follows:

i. Identity-based signature module
ii. Security center
iii. Application zone
iv. Manage the generation process of flow rules permissions for the valid applications.
v. Verify the flow rules validity.
vi. Authenticate the identities of the applications which generate flow rules insertion requests.
vii. Monitor the anomalous behaviors in SDN network conducted by malicious applications.

Then authors analyze, simulate and evaluate the proposed scheme by extending the Floodlight controller source code to support the functionality of PERM-GUARD.

Yustus et al. [20] proposed Northbound API security scheme to ensure the secure implementation of the REST NBI in SD controller, the scheme is designed to support the authentication and authorization and to capable to respond two questions:

i. Who are you?
ii. Do you have permission to access this network?

This scheme is using a token-based authentication and authorization process. The scheme was designed based on the OAuth 2.0 protocol [21].

Diogo et al. [22] proposed AuthFlow mechanism to guarantee the security between the hosts and servers in the SDN environment. This mechanism uses layer 2 protocols for authentication procedure. AuthFlow uses RADIUS authentication server which verifies the authenticity between the hosts and servers. The messages which are being sent between the RADIUS server and hosts are encapsulated by Extensible Authentication Protocol (EAP). To translate these messages from IEEE 802.1X to RADIUS packets is the responsibility of the authenticator server.

Hamza et al. [23] demonstrated the SDN OpenFlow controller security issues. They associated every security issue with existing countermeasures along with the description of these countermeasures. This demonstration concentrated to cover the security threats which may make the network service unavailable like DoS attack and some other attacks like (host hijacking attack, tampering attack, spoofing attack and man-in-the-middle attack). This research helps for further research to enhance the security of SDN OpenFlow controller.

Hong et al. [24] proposed host location hijacking attack and link fabrication attack which are attacking the OpenFlow network topology. The Floodlight controller is used to validate these two attacks. Mininet 2.0 simulation [25] is also used to demonstrate the topology of SDN network. These two attacks successfully poisoned the SDN topology. Therefore, the authors proposed TopoGuard security extension for SDN network topology using Floodlight controller. This extension protects the network from network poisoning attack.

3. Background and Motivation. To solve the issues of controller security, we used Zero- knowledge proof (ZKP). It is also known as Zero-Knowledge Protocol. It is a method occurs between two parties A and B where A is (the prover) tries to proof to B (the verifier) that he knows the secret X without exposing any other information about the secret or the secret itself. The proof procedure in ZKP is just knowledge about Knowledge, i.e. A will convince B that he knows the secret, B will be entirely confident that secret is true but will not learn anything as a result about this procedure, in this case, B gets zero-knowledge. Goldwasser, Micali and Rackoff introduced ZKP in [26]. To state that, the proving procedure is a ZKP, it must satisfy three properties:

i. Completeness: If the secret is true, the verifier will be convinced by the prover. Both prover and verifier must be honest and follow the protocol.
ii. Soundness: If the secret is false, the fake rover must not convince the honest verifier that, the secret is right.
iii. Zero-knowledge: If the secret is true, the verifier will learn nothing other than that, the secret is true.

The steps of ZKP occurs as follows: We assume that, there is A prover and B verifier, A wants to prove to B that he knows the secret$x$where $y = g^x$ without informing B what exact x is.

i. A generates a random number v and calculates $k = g^v$ and sends k to B.
ii. B generates a random number c as a challenge and sends it to A.
iii. A calculates $r = v - cx$ and sends it to B.
iv. B verifies $t = g^r * y^c$ if they are equal, then A is verified.

ZKP is the proof which revealed no information apart of the validity of the secret which the verifier demands the prover to proof. ZKP guarantees a high-level security protocol regardless of what verifier does, he will not learn any other information about the secret of the prover.

4. Proposed Scheme. The proposed scheme main concern is to secure the controller device in the SDN network from being accessed by malicious users. These malicious users try to control the whole SDN network, disable some services or shut the controller down by sending malicious requests to the controller. To protect the controller from this kind of malicious users, we preferred to use an identification scheme, so the user has to identify himself to the controller but not as a traditional way by sending his password in every attempt of login. The way we prefer is to let the user convince the controller that he knows the password without revealing the password itself or any partial information about it and prove that he is the real user. We utilized the ZKP identification scheme proposed by Feige, Fait and Shamir in [27] to improve the security of SDN controller against fake users. In this scheme, the user does not need to send his password in every login attempt where he has to prove that he knows the secret which controller has without revealing the actual secret or any information related to it. Three significant elements used in this scheme:

i. The controller: the verifier.
ii. The user: the prover.
iii. The Authentication Server: the trusted center which generates modulus n as a secret and has all user 's passwords and n numbers.

The user and controller have to agree on modulus n which is a product of two prime numbers, but no one will know the factorization of it. The module n will be assigned to both the user and the controller by the trusted center (the authentication server). After assigning the module $n$to both parties, the user has to prove to the controller that he knows the password without exposing the password itself. Thus both parties have to follow some steps to satisfy the ZKP. We divided these steps into four phases, User key generation phase, Registration, Login and Authentication phases.

4.1. User key-generation phase. In this phase, the user has to generate public and private keys as follows:

i. The user generates S random numbers S_1,.....,S_k in Z_n.
ii. The user generates I_j randomly and independently as $1/S_j^2$ (mod n).
iii. The user publishes $I = I_1$,.....,I_k as public keys and keeps $S = S_1$,.....,$S_k$as private keys.

4.2. Registration Phase. In this phase, the user has to create his credentials and register himself in SDN network to connect to the controller and get the network policies and rules from it, see Figure 4.1. Therefore, the user has to perform the following steps:

i. The user creates a user *ID (IDu)*, user password *(PW)* and public key I send them along with user IP address *(IPu)* to the controller as a registration request,ID_u//h(PW)//IP_u//$I_j$where h is a hash function to encrypt the password.
ii. The controller will forward the registration request to the authentication server to record user credential in the database.
iii. Authentication server stores the user credentials in its database and generates the modulus n (a product of two prime numbers) ($n = p * q$) where n is considered as a secret and assign it to both user and controller. Now, the user is registered in SDN network and has to login into it.

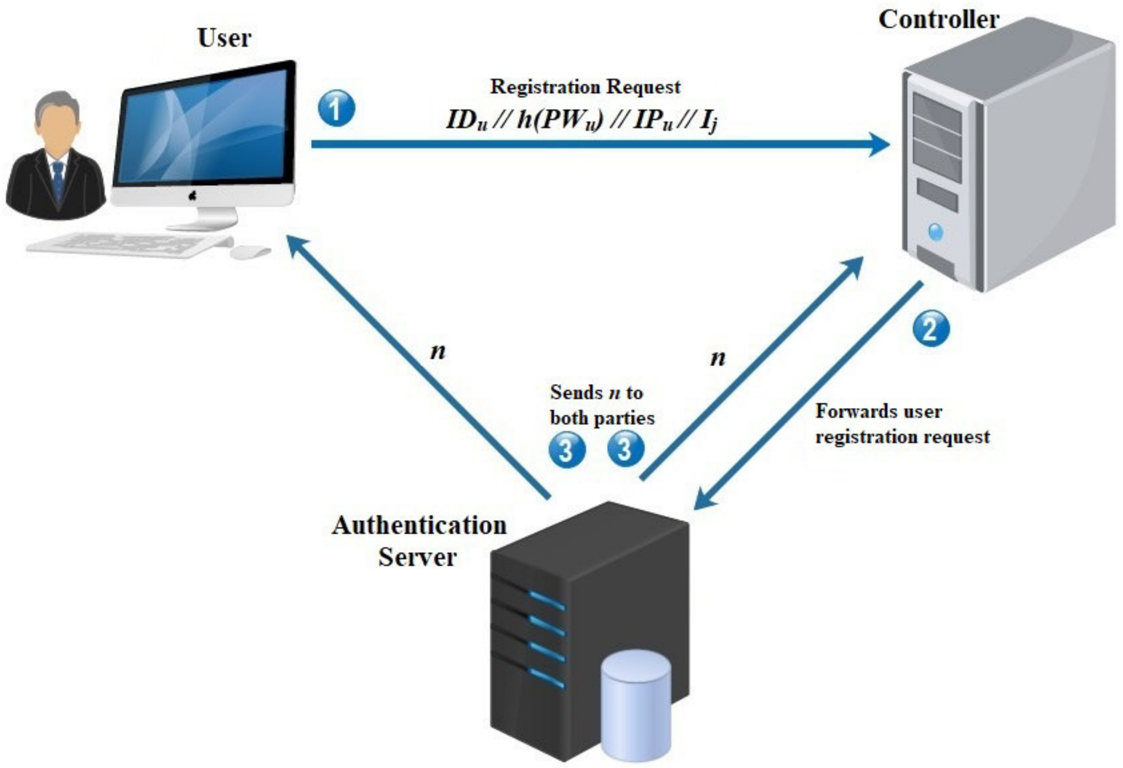

FIG. 4.1. *Registration Phase*

4.3. Login phase. In this phase, the user will try to login into SDN network by sending the controller a login request, see Figure 4.2. The user will not send the actual password instead he will use ZKP to prove his identity, and this occurs as the following steps:

1. The user chooses a random number$R$ and calculates $X = \pm R^2$ *(mod n)* then sends X to the controller.

2. The controller selects random numbers as Boolean vector $A_j = (A_1,.....,A_k\ )$ and sends them to the user as challenges.
3. The user calculates the value of $Y = R\prod A_{j=1}S_j\,(mod\ n)$ and sends Y to the controller.

Step 3 should be repeated in t iterations depends on the number of challenges in boolean vector $(A_1,.....,A_k)$. Suppose the controller sent 3 challenges in the boolean vector as $(A_1, A_2, A_3$ then the number of iteration in step 3 will be t=3. So the user should calculate this step 3 times as:

i. $Y = R\prod A_{1=1}S_1\ (mod\ n)$
ii. $Y = R\prod A_{2=1}S_2\ (mod\ n)$
iii. $Y = R\prod A_{3=1}S_3\ (mod\ n)$

where S_1, S_2, S_3 will be 3 private keys of the user.

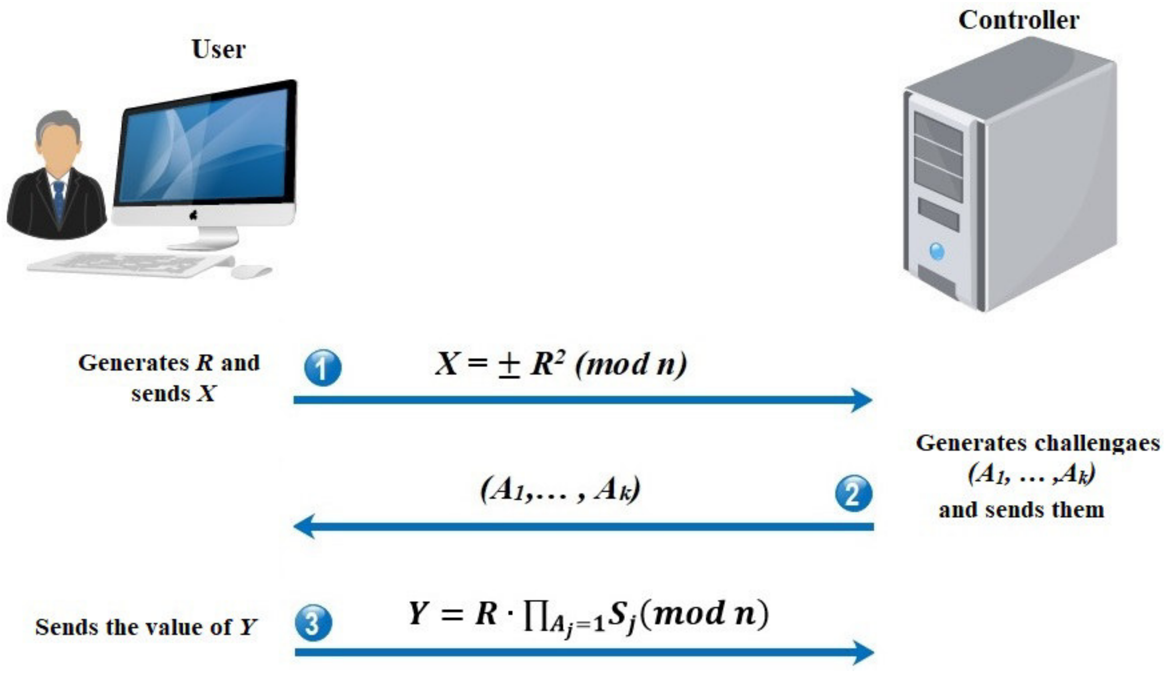
User
Controller
Generates R and sends X
1
$X = \pm R^2\ (mod\ n)$
Generates challengaes $(A_1, \ldots, A_k)$ and sends them
$(A_1, \ldots, A_k)$
2
Sends the value of Y
3
$Y = R \cdot \prod_{A_j=1} S_j(mod\ n)$

FIG. 4.2. *Login Phase*

Now, the user is waiting to the controller to accept his proof of identity to start the communication and exchange packets between them.

4.4. Identification phase. In this phase, the controller identifies the user proof of identity to allow him to send and receive the packets, see Figure 4.3. The identification step occurs as follow:

i. $X = \pm Y^2\prod A_{j=1}I_j\ (mod\ n)$ The identification step must be repeated in t iterations to accept the user proof of identity. The identification iterations are decided by the number of challenges in boolean vector $(A_1,.....,A_k)$ sent by the controller as we assumed that in login phase, the numbers of challenges are 3, then the number of the verification iterations t will be $t = 3$ and the calculation will be as follow:
ii. $X = \pm Y^2\prod A_{1=1}I_1\ (mod\ n)$
iii. $X = \pm Y^2\prod A_{2=1}I_2\ (mod\ n)$
iv. $X = \pm Y^2\prod A_{3=1}I_3\ (mod\ n)$

where $(I_1, I_2, I_3\ )$ will be 3 public keys of the user.

The controller accepts the user proof of identity if and only if the identification step is performed successfully in all t iterations otherwise the user proof of identity will be rejected. If the controller accepts the user's request, then the policies will be installed as entries into the flow table of the switch. Therefore the user will be able to communicate with network resources and other devices in SDN network. If the controller rejects the user's request, the user has to repeat the operation of login phase again with correct evidence.

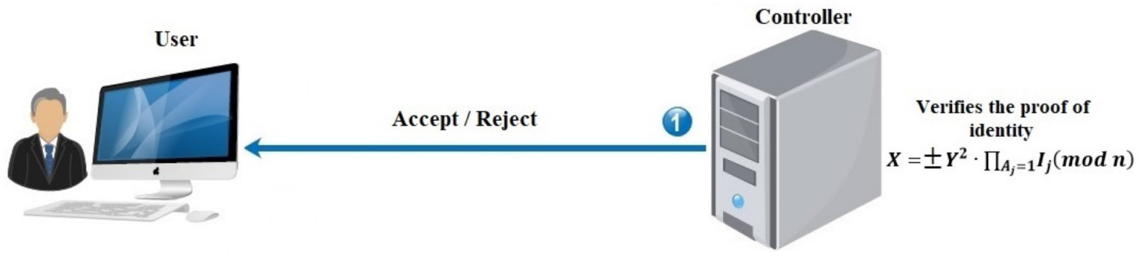
User
Controller
Accept / Reject
1
Verifies the proof of identity
$X = \pm Y^2 \cdot \prod_{A_j=1} I_j (mod\ n)$

FIG. 4.3. *Identification phase*

5. Result Discussion. In this section, the following parameters are used to evaluate the efficiency of the proposed scheme.

5.1. Computation Cost, Communication Cost and Storage Overhead Analysis. The primary goal of the proposed scheme is to reduce the overall computation cost, communication cost and storage overhead associated with the prover (user) and verifier (controller). To make the proposed scheme in practical use, the Table 5.2 along with notations Table 5.1, show that this scheme is an efficient scheme in term of computation cost communication cost and storage overhead for SDN environment

TABLE 5.1
Notations of Table 5.2

Symbol	Description
e	Exponent
m	Multiplication
mod	Modulus

TABLE 5.2
Computation cost, communication cost and storage overhead of login and identification phases

Phase	Computation Cost	Communication Cost/bit	Storage Overhead/bit
Login phase	$1e + 2m + 2mod$	2689	2560
Identification phase	$3e + 2m + 1mod$		

Many aspects must be taken into consideration to identify the efficiency of any identification scheme such as computation cost, communication cost and storage overhead. We mainly concentrate on the login and identification phases. These two phases are considered the major parts of the identification mechanism in this proposed scheme, and they are frequently executed as compared to other phases. Without loss of generality, the parameters *(R,I,S,A)* are considered to be 128-bit while the n is 1024-bit long. The computation cost is calculated according to how many mathematical operations occurred between user and controller sides in both login and identification phases. The communication cost focuses on the number of the bits are being sent between the user and controller in both login and identification phases. The calculation of the storage overhead occurs according to how many bits are the user and controller store in their devices. The parameters*(R,I,S,n)* are stored in the user side device, where*(A,n)* are stored in the controller side. The calculation of computational cost, communication cost and storage overhead in Table 5.2 is considered for one-time identification iteration.

5.2. Security Analysis. The proposed scheme is efficient to avoid various types of attacks as host impersonate attack, DoS attack and Man-in-the-middle attack due to its robust security mechanism of identifying the real users and invisibility of user credentials as well.

5.2.1. Host Impersonation Attack and Man-in-the-middle Attack. The proposed scheme can prevent the host impersonation attack as well as the man-in-the-middle attack by allowing the user to send his credential $ID_u//$h(PW)$//IP_u//I_j$ to the controller only once and that is in registration face. In the login phase, the user will send $X = \pm R^2$ *(mod n)* to the controller where$X$ changes in every attempt of login according to the secret n sent by the authentication server to both user and controller. In this case, the attacker cannot hack a consistent$X$ to impersonate the host to communicate the controller for subversive reasons.

5.2.2. DoS Attack. In the case of a DoS attack, the hacker will try to gain control over the controller to suspend some service or make the whole network unavailable. The proposed scheme prevents the issues of DoS attack by instructing the controller to verify whether the user knows the secret or not. If the user proofs that he knows the secret multiple times (which is a guarantee that the user is not a hacker) as mentioned in identification phase as $X = \pm Y^2 \prod (A_{j=1}) I_j$ *(mod n)*, then the user is able to communicate the controller.

6. Conclusion. The security enhancement of SDN has been taken seriously by researchers and enterprise networks in recent years especially the security between control and data planes. The users in the data plane have to prove their identity to the controller to gain access to it and get networking policies. In this paper, we presented the security features of the ZKP protocol and how the user identity proceed and proved. Then we proposed an efficient ZKP based identification scheme for securing SDN. This scheme guarantees that only real users can connect to the SDN controller. We also discussed the computation cost communication cost and storage overhead analysis to validate the efficiency of our proposed scheme along with security analysis.

REFERENCES

[1] M. Casado, T. Koponen, D. Moon, S. Shenker, *Rethinking Packet Forwarding Hardware,*In Proc. OfHotNets, 2008.

[2] ONF,*Software-Defined Networking: The New Norm forNetworks,*white paper, https://www.opennetworking.org.

[3] Xia, Wenfeng, Yonggang Wen, Chuan Heng Foh, Dusit Niyato, and Haiyong Xie, *A survey on software-defined networking.,*IEEE Communications Surveys & Tutorials 17.1 (2015): 27-51.

[4] Zhang, Yuan, Lin Cui, Wei Wang, and Yuxiang Zhang, *A survey on software defined networking with multiple controllers,* Journal of Network and Computer Applications (2017).

[5] Kreutz, Diego, Fernando MV Ramos, Paulo Esteves Verissimo, Christian Esteve Rothenberg, Siamak Azodolmolky, and Steve Uhlig, *"Software-defined networking: A comprehensive survey"* , Proceedings of the IEEE 103.1 (2015): 14-76.

[6] Bozakov, Zdravko, and Amr Rizk, *"Taming SDN controllers in heterogeneous hardware environments."* Software Defined Networks (EWSDN), 2013 Second European Workshop on. IEEE, 2013.

[7] Feamster, Nick, Jennifer Rexford, and Ellen Zegura. *"The road to SDN: an intellectual history of programmable networks."*ACM SIGCOMM Computer Communication Review 44.2 (2014): 87-98.

[8] Nunes, Bruno Astuto A., Marc Mendonca, Xuan-Nam Nguyen, Katia Obraczka, and Thierry Turletti., *"A survey of software-defined networking: Past, present, and future of programmable networks"* IEEE Communications Surveys & Tutorials 16.3 (2014): 1617-1634.

[9] OpenFlow Specification 1.0 http://www.openflow.org/documents/openflow-spec-v1.0.0.pdf., (Accessed: 2017/04/05) (2009). pp. 9196.

[10] OpenFlow Specification 1.1.0., http://www.openflow.org/documents/openflow-spec-v1.1.0.pdf., (Accessed: 2017/04/05) (2011).

[11] OpenFlowSpecification1.3.0., https://www.opennetworking.org/images/stories/downloads/specification/openflow-spec-v1.3.0.pdf. (Accessed: 2017/04/05) (2012).

[12] OpenFlow Specification 1.3.0., https://www.opennetworking.org/images/stories/downloads/specification/openflow-spec-v1.3.0.pdf.,(Accessed: 2017/04/05) (2012).

[13] OpenFlow Specification 1.4.0., https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.4.0.pdf, (Accessed: 2017/04/07) (2013).

[14] OpenFlow Specification 1.5.0., https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-switch-v1.5.0.noipr.pdf, (Accessed:2017/04/07) (2014).

[15] OpenFlow-Configuration Protocol 1.2, https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow-config/of-config-1.2.pdf, (Accessed:2017/04/07) (2014).pp. 101109.

[16] McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., et al *"OpenFlow: enabling innovation in campus networks."* ACM SIGCOMM Computer Communication Review 38.2 (2008): 69-74.

[17] Abdullaziz, Osamah Ibrahiem, Yu-Jia Chen, and Li-Chun Wang. *"Lightweight authentication mechanism for software defined network using information hiding."*Global Communications Conference (GLOBECOM), 2016 IEEE. IEEE, 2016.

[18] Tootoonchian, A., Gorbunov, S., Ganjali, Y., Casado, M., & Sherwood, R. *"On Controller Performance in Software-Defined Networks."*Hot-ICE 12 (2012): 1-6.

[19] Wang, M., Liu, J., Chen, J., Liu, X., & Mao*"Perm-guard: Authenticating the validity of flow rules in software defined networking."* Hot-ICE 12 (2012): 1-6.

[20] Oktian, Y. E., Lee, S., Lee, H., & Lam, J*"Secure your northbound SDN API."*Hot-ICE 12 (2012): 1-6.

[21] Jones, Michael, and Dick Hardt *The oauth 2.0 authorization framework: Bearer token usage.* No. RFC 6750. 2012.

[22] Mattos, Diogo Menezes Ferrazani, and Otto Carlos Muniz Bandeira Duarte. *"AuthFlow: authentication and access control mechanism for software defined networking."*Annals of Telecommunications 71.11-12 (2016): 607-615.

[23] Mutaher, Hamza, Pradeep Kumar, and Abdul Wahid. *"OPENFLOW CONTROLLER-BASED SDN: SECURITY ISSUES AND COUNTERMEASURES."* International Journal of Advanced Research in Computer Science 9.1 (2018).

[24] Hong, S., Xu, L., Wang, H., & Gu, G *"Poisoning Network Visibility in Software-Defined Networks: New Attacks and Countermeasures."* NDSS. Vol. 15. 2015.

[25] Mininet. Rapid prototyping for software defined networks.http://yuba.stanford.edu/foswiki/bin/view/OpenFlow/.

[26] Goldwasser, Shafi, Silvio Micali, and Charles Rackoff. *"The knowledge complexity of interactive proof systems."* SIAM Journal on computing 18.1 (1989): 186-208.

[27] Feige, Uriel, Amos Fiat, and Adi Shamir *"Zero-knowledge proofs of identity."* Journal of cryptology 1.2 (1988): 77-94.

Edited by: Khaleel Ahmad
Received: Nov 24, 2018
Accepted: Feb 11, 2019