ISSN 2303-4521
Periodicals of Engineering and Natural Sciences *Original Research*
Vol. 10, No. 2, April 2022, pp.231-240

Content delivery network for secure of software defined networking by using IPv4, OpenFlow, and ALTO

Nawfal Turki Obeis[1], Ahssan Ahmed Mohammed Lehmoud[2], Ahmed Fakhir Mutar[3,4]
[1] College of Information Technology, University of Babylon, Iraq
[2,4] Babylon Education Directorate, Iraq
[3]Al-Mustaqbal University College, Iraq

ABSTRACT

Software defined networking is a programmability function network by easiness for maintenance and configuration. The administrators of network can change the traffic rules during the commuting process. SDN is an arising network structure with programmability and centralization and this leads to introduce potential security concerns. Though the TLS ability support secure for control plane but computationally aggravating and complex to configure as well as not compatible with OpenFlow protocol. For this reason, a content delivery network can be used to increase the ability of network services dynamically and automatically. In order that relieve the threat we proposed architecture for SDN depending on CDN. In our proposed architecture, we use application layer traffic optimization (ALTO) protocol to be as servers enable mapping for the network to produce a summarized vision. We also hide the identity of the forwarding devices by take advantage of IPv4 and OpenFlow transaction identification fields into the control packets through implement of two authentication structures via efficient Salsa20 stream cipher. Finally, the work results explain the proposed architecture can efficiently eliminate of attack types and provide more detectability to attackers.

Keywords: SDN, Network, Authentication, Security, Protocol

Corresponding Author:

Nawfal Turki Obeis
Departement of Information Network
College of Information Technology, University of Babylon, Babylon, Iraq
nawfal.aljumaili@uobabylon.edu.iq

1. Introduction

CDN is short symbol of content delivery networks, it was suggested to deliver requests for services effectively for users requesting the service by assistance of content delivery network servers whose clients are the content provisioners [1]. Actually, from this network multiple was formed to control the problem of scalability of the boundaries of content provisioners to use multiple networks [1]. The following figure illustrate CDN structure

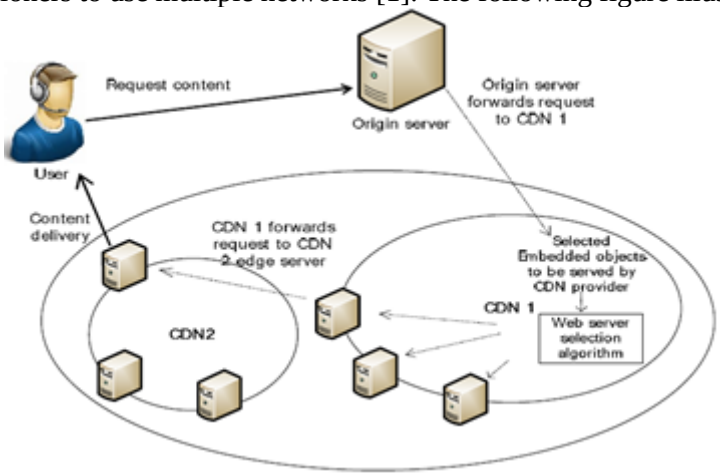

Figure 1. Content delivery networks [1]

In recent years, term appeared of SDN technology is the abbreviation of words of software defined network that is introduced to by limitation of controlling and programming the network with in an unequal controller point. The SDN is splatted to be control from the transmitting devices therefore facilitates and enhances network management. SDN is split up data plane from control plane of forwarding operation of network packets to make a central network. The SDN consists of three main planes, first one is a data plane, the second plane is an application plane and the last is control plane[2]. The following structure illustrate software defined network.

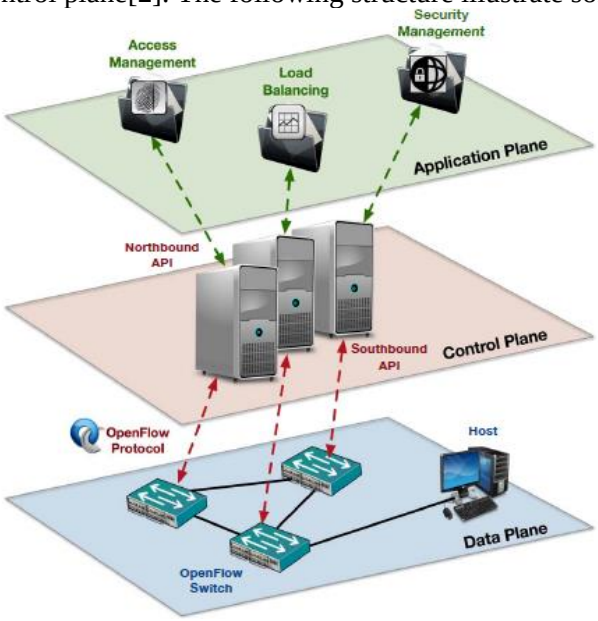

Figure 2. Software defined network[2]

The routing and security policies of SDN applications are taken to the control plane via allow of the northbound application programming interfaces. At the control plane, these policies are enforced on the forwarding devices in the SDN via the southbound application programming interfaces. So, the network flows can be controlled for the open southbound interfaces through the use of the first standard for the SDN, which is called OpenFlow. In other words, it is a protocol between the switches and controller for the purpose of sending and receiving[2].
An application layer traffic optimization (ALTO) is protocol implemented within SDN for provide incorporate in scope used like servers. ALTO is useful to functionalities in SDN to help for map full network for efficacious management[3]. ALTO include cost map and network map as a default maps[4]. SDN provide rendezvous services by controller plugged with ALTOserver through a well-defined interface[5]. Therefore, well use ALTO to creating a network by connected CDNi with SDN services to help for determine efficient routes [6-8].
For all of the above, a content delivery network can be used to increase the ability of network services dynamically and automatically. In order that relieve the threat we proposed architecture for SDN depending on CDN.
So, we will use application layer traffic optimization (ALTO) protocol to be as servers enable mapping for the network to produce a summarized vision.
We also hide the identity of the forwarding devices by take advantage of IPv4 and OpenFlow transaction identification fields into the control packets through implement of two authentication structures via efficient Salsa20 stream cipher.
Finally, the work results explain the proposed architecture can efficiently eliminate of attack types and provide more detectability to attackers.
The rest of our paper is organized as follows. Section 2 investigates a related works on SDN security for the last years. Section 3 provide the work on proposed architectures for strucure CDN with SDN by using IPv4, OpenFlow, and ALTO. Section 4 provide of discussion and analysis for work. Ultimately, we state our concluding remarks in Section 5.

2. Related work

There are so many works done based for provide security for SDN like (Ahmad et al., 2015), monitors network traffic and relies on the information stored in the control panel being constantly updated and focuses on sudden changes from suspicious and obfuscated movements to be detected and prevented and to identify the source of attacks[9].

(Shin et al., 2016) used three features of SDN to implement a set of security solutions, as the first method relies on security middleboxes in network infrastructure based on flow control rules installed bound or top in the controller[10].

(Dacier et al., 2017) relied on controlling the flow by creating a vision on the network. The idea is to create a knowledge of any element that enters the network and to make a central redirection decision to manage the network. Intelligence information can be collected through messages, requesting flow samples, and employing this information for security applications like IDS, IPS or DPI[11].

(Ajaeiya et al., 2017) used an effective security system by merging the Network-IDS with SDN networks and the mechanism was by allowing the terminal switches to transfer the runtime and network statistics in fixed time intervals as rather of parallel for the same controller[12].

(Tantar et al., 2018) mitigates potential attacks by relying on open source platforms and machine learning capabilities by prototyping the system through network state and behavior[13]. However, in our knowledge no one of many works implemented are concentrate to protect from the dangers of various types of attacks by using the method of communication with users in a secure manner and by achieving a variety of authentication for a single session.

3. Proposed architectures

SDN become more and more centralized this leads a possibility target for different types of attacks to purpose of force the important service points to be down. That's why we expand our proposed work for provide for the use of a mechanism to prevent an external attacks at the control level of the software-defined networking. The proposed system is lightweight as it requires simple operations in terms of calculation to hide the identity of the device in the control package and in the head of package in particular. In other words, we will divide our proposed work into two parts. The first part is proposed structure for IPv4 header identification with included of CDN to provide secure SDN by used ALTO services for summarize the network. And, the second part is structure for transaction identification field of the OpenFlow header and. The proposed work aim to security centralized system consisting of CDN and SDN based on IP header identification with ALTO services and OpenFlow header. Figure 3 will illustrate the architectures of proposed work

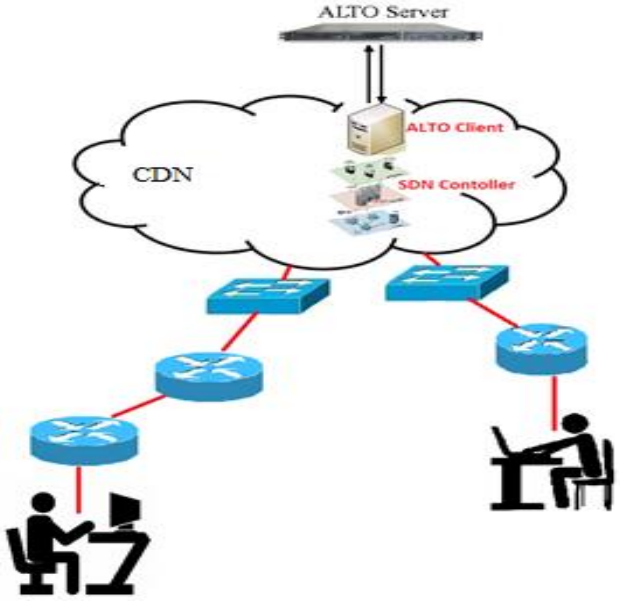

Figure 3. Architectures of proposed work

3.1 Proposed structure for IPv4 header identification

The IPv4 header identification architecture was created to be used by their own platform, through which randomization is implemented when the IP is generated. The proposed work differs from the previous methods in that it does not neglect the change from what is inserted in the IP distribution process. Further, the proposed structure also works to overcome the shortcomings found in most methods, as the network's MTU does not work except when it is pre-set.

The process of encryption and decryption of the IPv4 header identification in our proposed structure goes through three stages are production of initial kernel based on 64-bits block, then 12-bits utilized for confusion stream process and in the last is distribution reformative function. The proposed structure required introduction

key with length 256-bits with two sub-block each one with 64-bits for counter and initial kernel production. The production of both functions will mapped by 12-bits of IP base amount. In the final the control packet will be sent after the IP header identification has been distributed from the 12-bits base IP header identification to a private distribution-compatible 16-bits IP header identification for transmission.

1- Initial kernel production (IKP): It is a very important security procedure that was used in our work to maintain the authentication process. It was depend for Salsa20 algorithm for perform stream of key in high speed of stram cipher. Despite the aforementioned special uses of the Salsa20 for the purpose of stream cipher encryption, but it is possible to use it as a function for the production of initial kernel. The following figure illustrate Salsa20 scheme

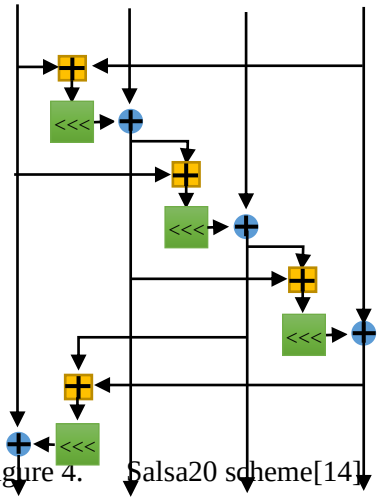

Figure 4. Salsa20 scheme[14]

Salsa20 algorithm is design based on pseudorandom operation instituted on ARX functions, as 32 bits addition function, then XOR function and in the last rotation function. The essence operation maps a 256 bits of key, 64 bits counter to a 512 bits block of the key stream, and 64 bits nonce. inwardly, salsa20 utilizes exclusive OR ⊕ , 32 bits addition mod 232 ⊞, and rotation (<<< shifting function) as constant distanceon an interior case of 16 words(each ward 32 bits). utilizing ARX functions obviates the potential of timing attacks in system achievements. The input 16 words are schudeling to be a 4×4 matrix[14].

Table 1. 16 words of Salsa20

“Exp1”	**SK**	**SK**	**SK**
SK	"nd4"	Nonce	Nonce
SP	SP	"3-by"	SK
SK	SK	SK	"mk"

The initial matrix is divided in to 8 words as key (SK), 2 words as a stream position (SP), 2 words of none (basically bits of additional stream position), and 4 steady words. For example in ASCII("Exp1" , "nd4", "3-by", and "mk") The essence function in algorithm is the quarter round (x, y, z, v) that selects a 4 words as input and 4 words as output produces[15].

$$y \mathrel{\wedge}= (x + v) <<< 7 \quad (1)$$
$$z \mathrel{\wedge}= (y + x) <<< 9 \quad (2)$$
$$v \mathrel{\wedge}= (z + y) <<< 13 \quad (3)$$
$$x \mathrel{\wedge}= (v + z) <<< 18 \quad (4)$$

So, 2 sequential rounds (row and column round) with each other are double round called it.
When we run the Salsa20 with a quad matrix based on particular rounds number, it will add the updated case to the matrix to produce a 64-byte for key. The proposed work for the IP header identification will take 12 bits of product key to be implemented in the second stage, which is confusion stream operation.

2- Confusion stream (CS): it is a permutation operation to product security for the input bits as well as randomness stream. The input bits from the previous stage are transposed according to a permutation

table to get 12 bits out it can be used in sequence matching table. As such, the security of permutation function product robustly based to a randomness for generation secret key. It's a permutation table, must not detect any idea for attacks on the device ID. The following fig illustrate permutation map.

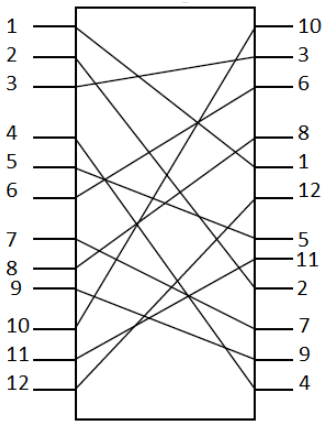

Figure 5. Confusion stream permutation map

Thereafter, matching table is utilizes as lookup table to matching and produce a 12 bits, that is substitute by least significant bits for the IP identification, It is indicated as the basic IP value of the new IP. So, matching table includes of 4096 insertion, everyone with 12 bits. In this case, any entry in the matching table will associated with a singular decimal position value, will be represented by 12 bits of a produce of new IP. In our work, indexing was used instead of traditional matching to speed up the process of mapping the new IP, Here it will be a constant time during indexing being a single access to memory (ie the interval needed to locate the stream of bits). The process of creating the matching table, which is stored and used in the indexing process, is done using the permutation to form the new IP.

3- Distribution reformative function(DRF) : Here is the third stage of work, which is a complementary function to the previous stages initial kernel production of creating the basic IP and confusion stream matching, and it is combined for generating the IP identification. The produce IP identification values will respond to the IP distribution of the implementation OS. As well as to guarantee high anti-discoverability and unattached of the hidden information. The third stage is established on UNIX IP identification production. The fact that UNIX-based systems are exposed to different distributions from the regular distribution because the queue is findable, which increases the predictability of the IP and IP collision averted[16]. Distribution reformative function is proposed as compliantly of a selector for IP identification values. The remaining four most significant bit of the whole IP identification area are used to product sixteen IP identification candidates to prohibit possibility to reuse the same IP within a findability queue.

In the last we proposed new map based on ALTO called (IP map). IP map will include a combination of the output IP from first part of our proposed system as IP encrypted values. These map will allow system to check packets from sent to confirm whether the packet is legitimate or spoofed. ALTO server will extract IP information from the packet by following steps.

a. When received packet ALTO will extract IP combination by decryption function.
b. Match the IP identification with their IP addresses
c. Traced back of the spoofed packets and then blocked it.
d. Distribution of related information of attacker to the interconnected CDNs by the SND.

The findable queue is used to storage newly IPs used. The queue length as 2^{12}, to be provides a perfect trade-off between non-recurrences with randomness during putting performance at account[16]. The following proposed structure within algorithm for encryption function of IP identification in our paper.

Algorithm
Input : counter, nonce, key, matching table, 12-bit device ID, IP table
Output : new IP ready to use
Step 1 Implement Salsa20 algorithm (Counter, Nonce, and Key)
Step 2 Perform quarter rounds for Salsa20 matrix

Step 3 Produce 12 bits stream by Salsa20
Step 4 Put 12 bits as MSB
Step 5 Implement confusion stream permutation to the 12 bits
Step 6 Select 4 bits sequence k for IP
Step 7 While selected IP in queue do select another IP
Step 8 Append selected IP to queue to obtain 16 IP candidates
Step 9 Correct the selected IP production from the 16 IP candidates
Step 10 Check if the selected IP is lately utilized. Means, selected IP out of queue.
Step 11 If queue length exceeds 4096 then extract the first IP in queue
Step 12 Return IP
The decryption function confirms the similarity of work but in reversing for encryption function. Figure 6 shows the IP identification through in our work for encryption and decryption functions.

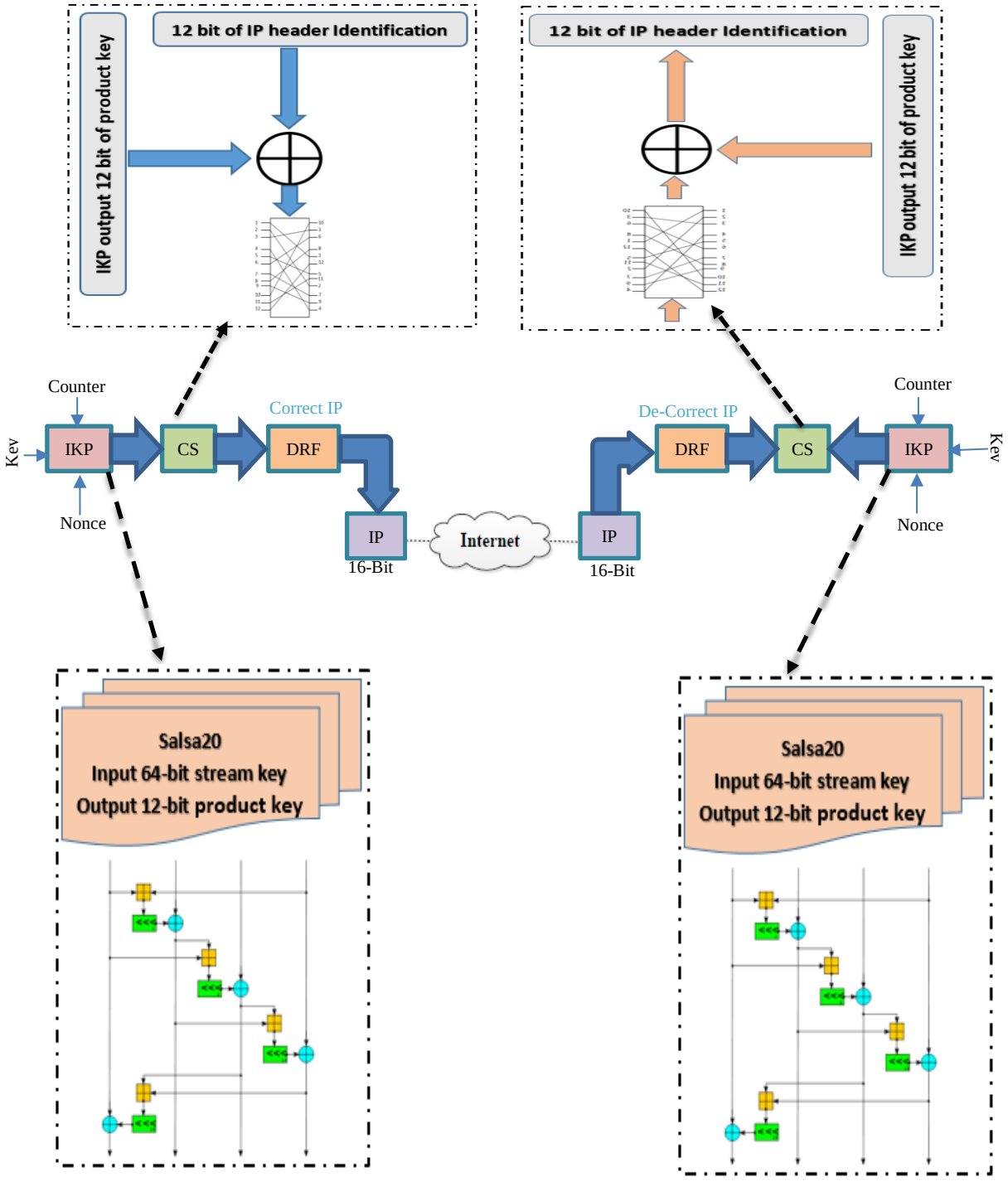

Figure 6. IP identification in encryption and decryption

3.2. Transaction identification field of the Openflow header

In such structure, it is possible to provide same field authentication to the Openflow header. To our understanding, the proposed work in this paper is the first algorithm in simplifying package pairing and supplying side channel communication. The main purpose of using the OpenFlow protocol is to match synchronous control messages using the 32-bit header field for transaction identifier.

Here we proposed the same mechanism used in IP identification but with selecting 32 bits from the output of the Salsa20 algorithm. Subsequently, using 32 bits in the confusion stream function to be permutation it to be used in the header of Openflow header for the control channel packet. In Openflow header identification will utilized permutation that was used in the IP identification but within 32 bits permutation matrix. In same time, well ignore the third stage of IP identification, the purpose of ignore distribution reformative function is to hold over the storage cost as well as computation as minimum as possible. Actually, ignore the distribution reformative function doesn't reduce the safety of the proposed method because 32 bits used is long enough to achieve security. The following structure is spatially for Identification field of the Openflow header in both of function.

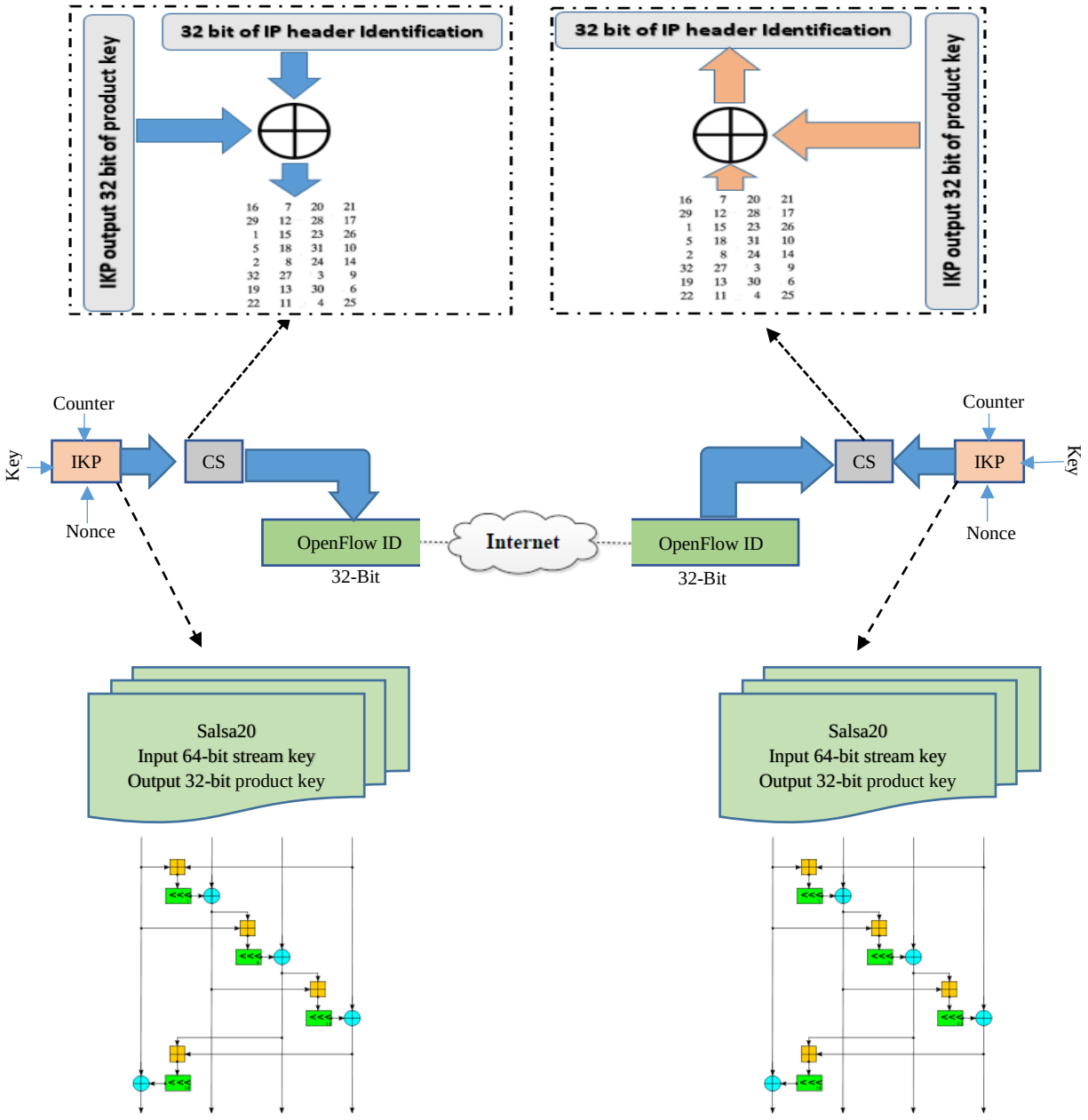

Figure 7. Transaction identification field of the Openflow header in encryption and decryption

Provide sufficient detail to allow the work to be reproduced. Methods already published should be indicated by a reference: only relevant modifications should be described.

4. Discussion and analysis

Through real testbeds, in this part well check the performance to our proposed work in terms of its resistance to multiple attacks. The proposed work is provide authentication between server and client during communication. IP identification with Openflow header identification are performed in a secure environment, without any malicious attacks.

We use controller benchmarking as a tool to simulate the OpenFlow switches number for evaluate controller during execution as simulation of DoS attack. In other word, used controller benchmarking as criterion based for throughput, response time, and latency. So, we used controller benchmarking with eight emulated switches in throughput mode. The benchmarking is performed to calculate the average throughput during execution and in the similarity steps for throughput under for the DoS attack. So, for elimination of the threat made by DoS attacks, we used CDN based on SDN to be transforming the network structure from a centralized network to be distributed SDN within CDNi.

Through the strategy used in our work, the spoofing attack can be controlled because spoofing attack uses IP spoofing of the SDN network. The spoofing attack hides malicious packets as legitimate packets via a fake IP while blaming other innocent users for its malicious actions. In our proposed scheme with IP identification mode well provide secure communication by detects and blocks spoofing attack.

Resource exhaustion attack is based on draining resources like energy, bandwidth of network, processing power, and storage capacity through service requests or the use of communications, and through the proposed work strategy, network resources and energy will be preserved as well as prevent this type of attack and similar attacks like (distributed denial of service and battery draining attacks).

Eavesdropping attack is often based on eavesdropping and stealing sensitive information sent through the network in a manner called data sniffing. The proposed work provides a strong security method for communication that prevents such an attack.

For smudge attack and shoulder surfing attack, when generate 32 bits as stream key it is sufficient to beat the attack.

For simulation results for our proposed work will present table 2 to illustrate a number of six tests everyone is sending ten network requests to the server via the SDN within CDNi Structure. The results in table contains our proposed work implementation hiding IP value and without hiding.

Table 2. Proposed work for hiding and without hiding via round trip time

Request_No	Round_trip_time	Test_1	Test_2	Test_3	Test_4	Test_5	Test_6
1	Without hiding	3.20	2.2	3.76	2.7	2.81	2.12
	With hiding	4.45	2.38	2.34	2.87	2.61	1.18
2	Without hiding	0.54	0.056	2.26	0.45	0.94	0.36
	With hiding	1.03	0.331	0.632	0.618	1.62	0.058
3	Without hiding	0.059	0.051	0.223	0.12	0.24	0.07
	With hiding	0.423	0.07	0.053	0.482	0.082	0.058
4	Without hiding	0.13	0.108	0.052	0.082	0.055	0.118
	With hiding	0.053	0.075	0.055	0.05	0.03	0.093
5	Without hiding	0.058	0.447	0.0113	0.18	0.088	0.036
	With hiding	0.055	0.445	0.057	0.052	0.053	0.056
6	Without hiding	0.043	0.032	0.051	0.045	0.13	0.037
	With hiding	0.167	0.407	0.084	0.236	0.448	0.123
7	Without hiding	0.037	0.038	0.067	0.083	0.052	0.038

Request_No	Round_trip_time	Test_1	Test_2	Test_3	Test_4	Test_5	Test_6
	With hiding	0.155	0.055	0.387	0.127	0.045	0.532
8	Without hiding	0.031	0.038	0.058	0.046	0.0345	0.0346
	With hiding	0.058	0.48	0.053	0.058	0.345	0.442
9	Without hiding	0.037	0.037	0.449	0.038	0.0496	0.037
	With hiding	0.048	0.482	0.058	0.456	0.057	0.124
10	Without hiding	0.013	0.037	0.057	0.058	0.052	0.036
	With hiding	0.493	0.746	0.045	0.057	0.057	0.058

The proposed work tests that illustrate the round trip time with a lower rate of 0.013 and an upper rate of 3.76 in millisecond. The round trip time is more decreases between the first packets to the last packet for all tests in the results for without hiding. In the second hand, illustrated similar results but with require a few extra time as 4.45 millisecond in the first test as an upper rate in the execution work. Throughout the rest of the results can be seen in the table, and the results are similar in both tests from the next packets and from the next test.
The first packet required extra time because first its bring rules from the controller of SDN that is relating with way to redirect it. This situation is not repeated for later packets, because the switches are learned from the rules of the packet forwarding packets. Therefore our proposed work for switches are not produce any important overhead that is almost negligible.

5. Conclusions

In the proposed work, we experimented the possible scenarios for different attacks that ability to implement versus SDN controller. It turns out that the used attacks drain controller resources through disclose of connected host within networked. After that, we proposed structure to secure SDN based in CDN has ability for safeguard the controller through hiding of forwarding devices identities into a control packets via efficient Salsa20 stream cipher algorithm. So, will detect spoofed IP and creates an identification combination of IP address especially to provided and saved through ALTO protocol to be used as a map in entire network. Therefore our work unique will produce a defense versus distributed or insider attacks to detection and prevention.

References

[1] B. Niven-Jenkins, F. L. Faucheur, and N. Bitar, "Content distribution network interconnection (CDNI) problem statement," *RFC6707, Sep*, 2012.

[2] K. Benton, L. J. Camp, and C. Small, "OpenFlow vulnerability assessment," in *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*, 2013, pp. 151-152.

[3] N. I. Mowla, I. Doh, and K. Chae, "An efficient defense mechanism for spoofed IP attack in SDN based CDNi," in *2015 International Conference on Information Networking (ICOIN)*, 2015, pp. 92-97.

[4] B. Jankowski, W. Mazurczyk, and K. Szczypiorski, "PadSteg: Introducing inter-protocol steganography," *Telecommunication Systems*, vol. 52, pp. 1101-1111, 2013.

[5] O. I. Abdullaziz, V. T. Goh, H.-C. Ling, and K. Wong, "AIPISteg: An active IP identification based steganographic method," *Journal of Network and Computer Applications*, vol. 63, pp. 150-158, 2016.

[6] M. Arumaithurai, J. Seedorf, G. Paragliela, M. Pilarski, and S. Niccolini, "Evaluation of ALTO-enhanced request routing for CDN interconnection," in *2013 IEEE International Conference on Communications (ICC)*, 2013, pp. 3519-3524.

[7] M. Shibuya, Y. Hei, and T. Ogishi, "ISP-friendly peer selection mechanism with ALTO-like server," in *2011 13th Asia-Pacific Network Operations and Management Symposium*, 2011, pp. 1-8.

[8] S. T. Zargar, J. Joshi, and D. Tipper, "A survey of defense mechanisms against distributed denial of service (DDoS) flooding attacks," *IEEE communications surveys & tutorials*, vol. 15, pp. 2046-2069, 2013.

[9] I. Ahmad, S. Namal, M. Ylianttila, and A. Gurtov, "Security in software defined networks: A survey," *IEEE Communications Surveys & Tutorials*, vol. 17, pp. 2317-2346, 2015.

[10] S. Shin, L. Xu, S. Hong, and G. Gu, "Enhancing network security through software defined networking (SDN)," in *2016 25th international conference on computer communication and networks (ICCCN)*, 2016, pp. 1-9.

[11] M. C. Dacier, H. König, R. Cwalinski, F. Kargl, and S. Dietrich, "Security challenges and opportunities of software-defined networking," *IEEE Security & Privacy*, vol. 15, pp. 96-100, 2017.

[12] G. A. Ajaeiya, N. Adalian, I. H. Elhajj, A. Kayssi, and A. Chehab, "Flow-based intrusion detection system for SDN," in *2017 IEEE Symposium on Computers and Communications (ISCC)*, 2017, pp. 787-793.

[13] T. Engel, "On Using Cognition for Anomaly Detection in SDN," *EVOLVE-A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation VI*, vol. 674, p. 67, 2017.

[14] V. K. Gurbani, M. Scharf, T. Lakshman, V. Hilt, and E. Marocco, "Abstracting network state in Software Defined Networks (SDN) for rendezvous services," in *2012 IEEE international conference on communications (ICC)*, 2012, pp. 6627-6632.

[15] D. J. Bernstein, T. Chou, and P. Schwabe, "McBits: fast constant-time code-based cryptography," in *International Conference on Cryptographic Hardware and Embedded Systems*, 2013, pp. 250-272.

[16] K. Deepthi Kakumani, K. Singh, and S. Karthika, "Improved related-cipher attack on Salsa and ChaCha: revisited," *International Journal of Information Technology*, pp. 1-8, 2022.