Performance Evaluation of Software Routers with VPN Features

Hasan Redžović, *Graduate Student Member, IEEE,* Aleksandra Smiljanić, *Member, IEEE,* and Bogdan Savić

***Abstract* - This paper presents implementation and analysis of the VPN software router which is based on Quagga and strongSwan open-source software tools. We validated the functionalities of strongSwan and Quagga in realistic environment which include scenarios with link failures. Also, we measured and analyzed the performance of encryption and hash algorithms supported by strongSwan software, in order to advise an optimal VPN configuration that provides the best performance.**

***Keywords* - IPsec protocol, Quagga, Software Routers, Software VPN Solutions, strongSwan.**

I. Introduction

THE performance of commodity PC hardware is constantly improving. New affordable hardware components such as faster CPUs with a larger number of cores and higher speed network cards are being released each year. These constant improvements of commodity hardware provide new opportunities for utilization of software routers. Flexibility of development, implementation and configuration are the main advantages of software defined networks (SDN) [1] and software routers [2].

The most CPU consuming networking features are related to security, in particular to encryption and hash algorithms. In this paper, we examine data plane capabilities of software router with VPN features that is based on the Linux kernel network stack. The data plane that implements IP packet routing and VPN functionality in the Linux kernel space can be used as a foundation for creating SDN with VPN features.

Using Quagga [3] and strongSwan [4] software tools, we have created the software router with VPN features and analyzed its performance in real conditions. Quagga is a popular open-source control plane that utilizes Linux network stack as a router data plane. VPNs can be implemented using various security protocols such as IPsec, TLS/SSL and SSH that operate at different layers of the TCP/IP stack [5].

SSH protocol protects data at the application layer and it can be used in port-forwarding mode to create secure tunnels for data exchange between applications. TLS/SSL provides security at the transport layer. IPsec protocol provides protection at the network layer and enables creation of secure tunnels for the exchange of all IP packet traffic between distant private networks or users [6]. IPsec protects not only data exchanged between the users, but also their identities.

Software-based IPsec can use Linux kernel IPsec features or it can be implemented in the user space. We have shown earlier that IPsec implemented in the user space has a poorer performance than the IPsec implemented in Linux kernel [7].

StrongSwan implements IPsec protocol in Linux kernel. Using IKE daemon strongSwan configures ESP and AH protocols implemented in Linux kernel. We integrated strongSwan with Quagga and analyzed VPNs robustness to changes of network topology. StrongSwan is a flexible software VPN solution that supports a large number of encryption and hash algorithms. We have analyzed the performance of encryption and hash algorithms provided by strongSwan, in order to provide insight into capabilities of strongSwan and determine optimal VPN configurations.

In the rest of this paper, Section II describes briefly Quagga software and its architecture. Section III presents the basics of the IPsec protocol. The strongSwan software with the list of supported security algorithms is described in Section IV. In Section V, we analyze the functionalities of Quagga and strongSwan in real networking environment. Section VI presents evaluation of encryption and hash algorithms in strongSwan. Finally, Section VII discusses results and our plans for future work.

II. Quagga

Time critical applications can be highly optimized in kernel. However, user space provides more stable and flexible environment for applications than kernel. Quagga architecture utilizes flexible user space environment for implementing control plane. Also, Quagga uses Linux kernel network stack as time critical data plane. Quagga

Paper received July 3, 2017; revised October 26, 2017; accepted October 30, 2017. Date of publication December 25, 2017. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Grozdan Petrović.

This paper is a revised and expanded version of the paper presented at the 24th Telecommunications Forum TELFOR 2016 [11].

This work was supported by the Serbian Ministry of Science and Education (project TR-32022), and companies Telekom Srbija, and Informatika.

Hasan Redžović is with the Innovation center of School of Electrical Engineering, University of Belgrade, Bul. kralja Aleksandra 73, 11120 Belgrade, Serbia (phone: 381-64-4641615; e-mail: hasanetf@live.com).

Aleksandra Smiljanić is with the School of Electrical Engineering, University of Belgrade, Bul. kralja Aleksandra 73, 11120 Belgrade, Serbia (e-mail: aleksandra@etf.rs).

Bogdan Savić is with the School of Electrical Engineering, University of Belgrade, Bul. kralja Aleksandra 73, 11120 Belgrade, Serbia (e-mail: bogdan.savic1503@gmail.com).

supports various routing protocols such as RIP, OSPF, BGP and IS-IS. Using independent process (daemon) called Zebra, Quagga connects the control plane to the data plane, and integrates software router on commodity hardware in this way. Fig. 1 illustrates the Quagga architecture [3].

Each routing protocol of control plane works as a daemon. The main daemon Zebra controls all these daemons, and updates the routing table in kernel when the network topology changes based on the information obtained from the daemons.

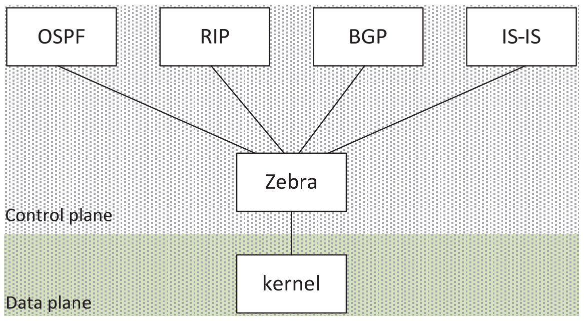

Fig. 1. Quagga architecture [3].

III. IPSEC PROTOCOL

IPsec protocol provides data protection at the network layer which means that it will protect all IP packets between distant private networks or users. IPsec protocol is composed of three protocols with different security roles and features:

- Internet Key Exchange (IKE) - handles creation of IPsec tunnels and authentication of distant peers in VPN. IKE protocol also provides periodical symmetrical key exchange;
- Authentication Header (AH) - provides IP packet integrity protection;
- Encapsulating Security Payload (ESP) - provides IP packet confidentiality and integrity protection.

Fig. 2 shows IP packet encapsulation with ESP protocol in tunnel mode. Original IP packet is encrypted with various encryption algorithms such as AES and 3DES which guarantee data confidentiality. ESP header and ESP trailer added to IP packet contain information about IP packet and established IPsec tunnel. Integrity protection of ESP packet is handled by a hash algorithm, such as MD5, SHA1 and SH2, which creates a fixed size hash unique for every packet. Finally, ESP packet gets an additional IP header which is used for sending ESP packet between two VPN gateways or peers through unsecured part of the network.

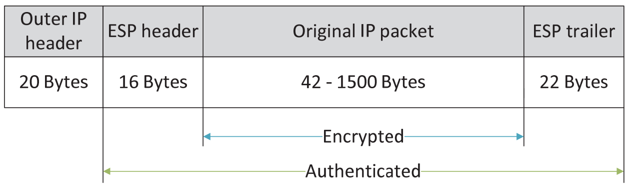

Fig. 2. IP packet encapsulation with ESP protocol in tunnel mode.

IPsec protocol supports many different and experimental encryption and hash algorithms with different levels of security. Level of security is measured as a number of compute operations necessary to decrypt protected data with brute force.

IV. STRONGSWAN

StrongSwan is open-source software that implements IPsec based VPN solutions using Linux, FreeBSD, OS X or Android operating system. Encryption and hash algorithms require intensive computing and, for this reason, ESP and AH protocols are implemented in kernel. On the other side, IKE protocol is implemented in user space, and it utilizes ESP and AH protocol kernel support. IKE protocol is implemented as an independent daemon named Charon which communicates with different peers in network using IKE protocol messages. After connection establishment, authentication and agreement on IPsec tunnel encryption and hash algorithms, daemon Charon configures ESP and AH protocols in kernel space. Packet processing path is always in kernel space. StrongSwan supports the following list of encryption and hash algorithms:

- Encryption algorithms: 3DES (rfc1851), AES (rfc3962), CAST-128 (rfc2144), Blowfish [8], Camellia (rfc3713);
- Hash algorithms: MD5 (rfc1321), SHA-1 (rfc3174), SHA-2 (rfc6668), AES XCBC (rfc3566), AES CMAC (rfc4493);
- Encryption algorithms with integrated hash algorithms: AES CCM (rfc5084), AES GCM (rfc5084) AES GMAC (rfc4543), ChaCha20Poly1305 (rfc7539).

V. EVALUATION OF VPN CONFIGURATIONS

Common IPsec VPN network configuration is based on VPN gateways which separate a safe private network domain from an unsafe public network. The main function of VPN IPsec gateway is to connect remote private networks and/or users by creating secure IPsec tunnels. Fig. 3(a) illustrates the basic method of connecting VPN gateway with a public network. Figs. 3(b) and 3(c) show cases with redundant links which improve VPN resilience to link failures. Redundant links can be installed between VPN gateway and public network access router, Fig. 3(b), and between VPN gateway and another Internet Service Provider (ISP) as shown in Fig. 3(c). The configurations with redundant VPN links are not fully supported by strongSwan and other similar software VPN solutions. Usually, special scripts and custom changes in programing code are needed in order to configure strongSwan to support VPN with redundant links. Also, these solutions are often not flexible enough to successfully resolve a high number of changes in network topology.

By combining strongSwan VPN gateway and Quagga software router, we manage to configure VPNs which can adapt to the network topology changes when the link failures happen. We will demonstrate behavior of the implemented software router with VPNs in the case of failures. The testing environment shown in Fig. 4

comprises five software routers connected with 1 Gbit/s links. All software routers in Fig. 4 are using the OSPF routing protocol in Quagga. StrongSwan was installed on the routers R1 and R4. IPsec tunnel has been configured between two private networks: *net_1* and *net_2* using ESP protocol in tunnel mode. The path of the created IPsec tunnel is marked with green color in Fig. 4 and includes routers R1, R2, R3 and R4. Fig. 4 also shows the path of ESP packets through the router R1 data plane.

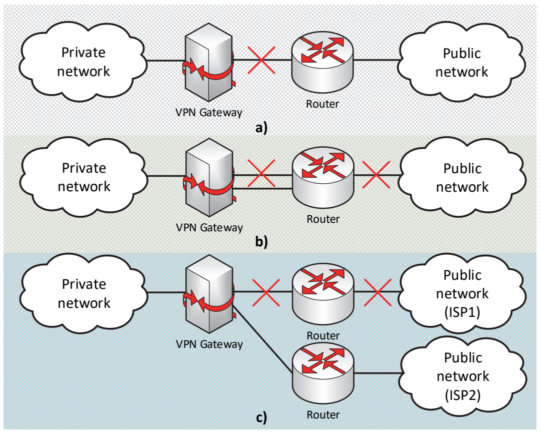

Fig. 3. a) Common VPN configuration; b) and c) VPN configuration with redundant links.

The numerated arrows in data plane of the router R1 represent the processing sequence for ESP packets received from router R4 and network *net_2*. This sequence has the following order:

- Router R1 is receiving ESP packets through the eth3 port and the IP lookup is performed using IP address of the outer IP header shown in Fig. 2;
- It is determined that ESP packets have the IP address of the R1 eth3 port;
- ESP packets are passed to strongSwan for decapsulation. Using ESP protocol in kernel space, strongSwan decapsulates IP packets sent from net_2;
- New IP lookup is performed using IP address of the original IP packet sent from network net_2;
- It is determined that IP packet has the IP address of network net_1. Then, IP packet is passed to the eth1 port of R1 for further delivery through network net_1.

Using OSPF protocol messages, routers communicate with each other in order to learn network topology and determine the shortest path to each node in network. Then, routers update their routing tables. If the network topology changes, for example a new link is added or a link fails, each router in network is notified and the routing table is updated based on the shortest path Dijkstra algorithm. In the test shown in Fig. 5, link between routers R1 and R2 is disabled. Using OSFP protocol, all routers have updated their information on network topology, calculated the shortest path to each node in network and, finally, updated their routing tables.

Before disabling the link, router R1 was using the eth3 port to communicate with router R2. Also, strongSwan configuration files for routers R1 and R4, use the IP address of the eth3 port as the IPsec tunnel endpoint. When IKE daemon Charon initiates establishment of an IPsec tunnel, IP addresses of the IPsec tunnel endpoints are necessary for exchanging IKE protocol messages. After disabling the link between routers R1 and R2, the OSPF protocol on router R1 is still broadcasting IP address of the eth3 port to the rest of the network. Other routers (R2, R3, R4 and R5) can still communicate to the R1 eth3 port using a new path through the R1 eth2 port.

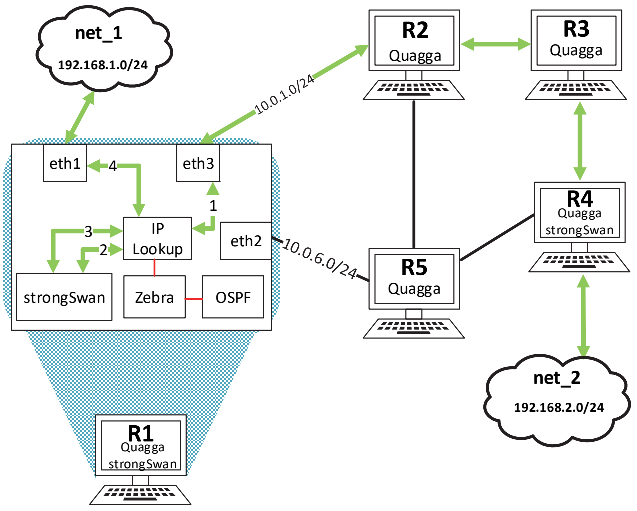

Fig. 4. IPsec tunnel path through kernel and public network.

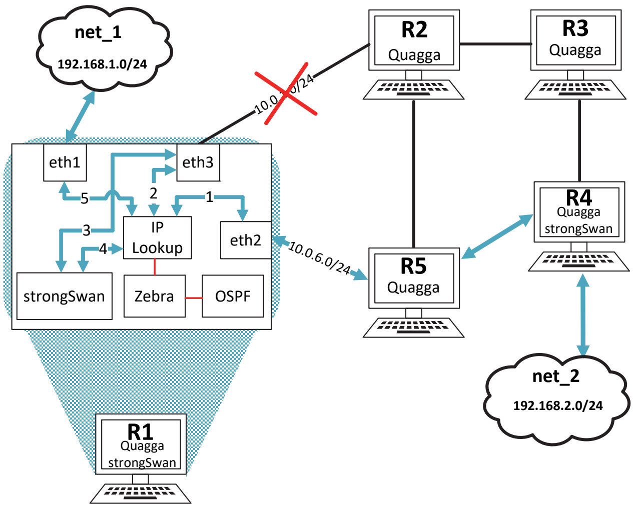

Fig. 5. IPsec tunnel through network and kernel after the link failure.

Unhindered communication between R1 eth3 port and the rest of the network allows automatic creation of the new IPsec tunnel between routers R1 and R4 along a different path shown in Fig. 5. In this case, ESP packets are exchanged between routers R1, R5 and R4, and their path is marked in Fig. 5 with blue color. The processing sequence at router R1 for ESP packets received from router R4 is also shown in Fig. 5 and it has the following order:

- Router R1 is receiving ESP packets through the eth2 port and the IP lookup is performed using the IP address of the outer IP header;
- It is determined that ESP packets have the IP address

of the R1 eth3 port;

- ESP packets are passed to strongSwan for decapsulation;
- The rest of the packet processing path is the same as in the first case with the link between routers R1 and R2.

Unlike in the cases shown in Figs. 3(b) and 3(c), which cannot be implemented using strongSwan, VPN gateway on router R1 can automatically switch IPsec tunnel to different paths thanks to the Quagga control plane.

VI. Performance of the IPsec security algorithms

Flexibility of software implementations enables fast and relatively easy development of new security algorithms. However, encryption algorithms are CPU intensive operations and it is necessary to analyze all strongSwan supported options to determine which algorithms provide the fastest packet processing. In order to find the best combination of strongSwan encryption and integrity protection algorithms, we conducted a large number of tests using testing environment shown in Fig. 6.

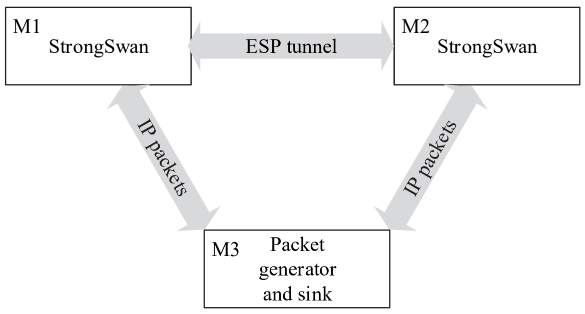

Fig. 6. Testing environment.

The testing environment comprises machines M1, M2 and M3 which were connected with 1 Gbit/s or 10 Gbit/s links. The performance evaluation of encryption and integrity algorithms was conducted using two different hardware configurations for machines M1, M2, and M3.

Both hardware configurations are shown in table 1 and they are labeled as a *hardware configuration A* and *hardware configuration B*. Hardware configuration A represents a low-cost commodity PC with a modest performance which uses 1 Gbit/s NICs. Hardware configuration B represents a high-end commodity PC which can use 10 Gbit/s NICs and it is intended for more advanced CPU intensive tasks. Hardware configurations A and B exemplify typical commodity PC machines that can be used for software based VPN implementations.

In testing environment shown in Fig. 6, the secure connection is established between machines M1 and M2 using ESP protocol in tunnel mode. Machine M3 simulates private networks and acts as a packet generator and sink for all decrypted and forwarded packets. Also, machine M3 is using a data generator based on data plane development kit (DPDK) [9] in order to provide maximal packet throughputs on links. DPDK packet generator guarantees a maximum packet load for machines M1 and M2. In testing environment shown in Fig. 6, packets are processed as follows: (i) IP packet is generated on machine M3 and sent to machine M1; (ii) one of the various VPN configurations is used to encrypt and encapsulate IP packet into ESP packet; (iii) ESP packet is sent to machine M2; (iv) machine M2 decrypts and decapsulates IP packet from ESP packet and sent it to machine M3; (v) machine M3 measures packet throughput of incoming IP packets. Simultaneously, the same packet processing is executed in the opposite direction (M3-M2-M1-M3).

For hardware configuration A which is using 1 Gbit/s links, machine M3 was generating 1.488 million packets per second (Mpps) in one direction. This maximal packet throughput is only achieved if a packet generator generates shortest IP packets, 64 bytes long. If the packet length increases, the maximal packet throughput decreases. In the case of hardware configuration B, which is using 10 Gbit/s links, DPDK packet generator is generating 14.88 Mpps in one direction. DPDK packet generator is sending packets in both directions, resulting in a total throughput of 2.976 Mpps for hardware configuration A and 29.76 Mpps for hardware configuration B.

Table 1: Hardware configurations for machines M1, M2 and M3.

Hardware configuration A	
CPU	Intel i5-3470 3.2GHz
RAM	16GB DDR3, 1600MHz
NIC	Intel 1 Gbit/s
Hardware configuration B	
CPU	Intel i7 3770k 3.9 GHz
RAM	16GB DDR3, 1600MHz
NIC	Intel 10 Gbit/s

For both hardware configurations A and B, the tests are divided into two groups. The first group of tests includes a combination of encryption algorithms (AES, Blowfish, Camellia, CAST-128 and 3DES) with different integrity algorithms (MD5, SHA1, SHA2, AES XCBC and AES CMAC). The second group of tests includes encryption algorithms that also provide integrity protection (AES CCM, AES GCM, AES GMAC and ChaCha20 Poly1305). For hardware configuration A, we used standard strongSwan distribution with default configuration which do not include all mentioned security algorithms. However, for hardware configuration B, we used the latest version of strongSwan with a custom configuration that enables all supported security algorithms. Various strongSwan distributions use relatively the same kernel modules for common encryption and integrity protection algorithms. Thus, in our tests, usage of different strongSwan versions didn't noticeably impact performance results.

Fig. 7 shows the performance results for separate encryption and integrity algorithms in the case of hardware configuration A. Different encryption algorithms (AES, Blowfish, Camellia, CAST-128 and 3DES) were combined with three hash algorithms (MD5, SHA1 and SHA2). Fig. 7 shows that the packet throughput depends on the encryption algorithm in a similar way for all hash algorithms under the consideration. Integrity algorithm MD5 is less CPU intensive and provides a better performance than SHA1 and SHA2_512, but MD5 also provides a less secure integrity check. Encryption algorithm AES provided the fastest packet processing in comparison to the other encryption algorithms. This performance advantage is due to the fact that strongSwan, Linux kernel and hardware support the Intel AES-NI [10] technology.

Fig. 8 shows performance results for the integrated encryption and integrity algorithms in the case of hardware configuration A. The standard StrongSwan distribution supports AES CMM (CBC MAC Mode), AES GCM (Galois/Counter Mode), AES GMAC (Galois Message Authentication Code) and ChaCha20 Poly1305 algorithms.

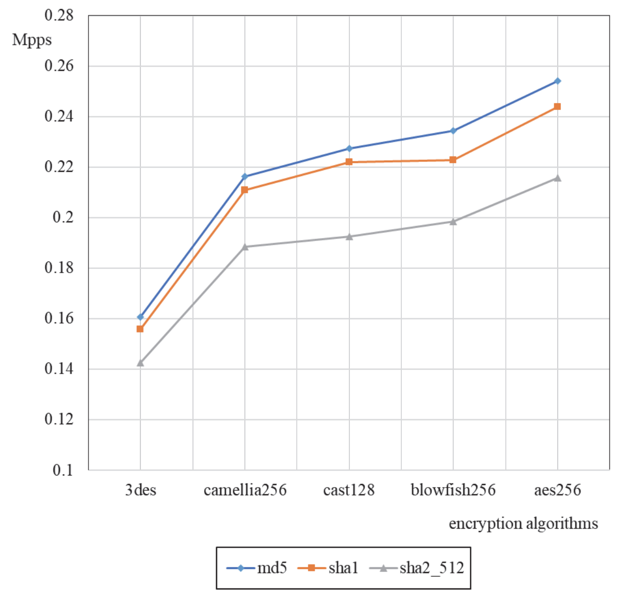

Fig. 7. Packet throughputs for separate encryption and integrity algorithms in hardware configuration A.

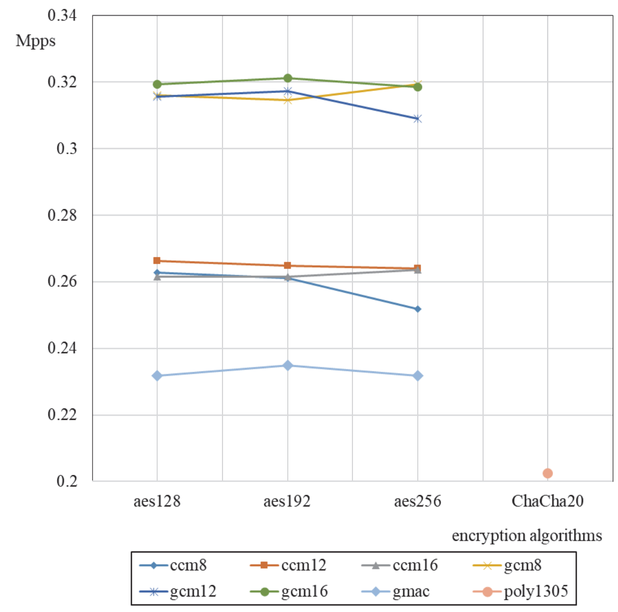

Fig. 8. Packet throughputs for the integrated encryption and integrity algorithms in hardware configuration A.

Fig. 8 also shows packet throughput for different encryption algorithms key lengths in order to analyze their influence on performance. The CMM mode which is used with AES (AES CMM) is a combination of algorithms based on the CBC (Cipher Block Chaining) and MAC (Message Authentication Code) architectures. Encryption key length did not notably affect performance. Security algorithm AES GCM provided the highest packet throughput in comparison to other tested security algorithms. The total packet throughput of forwarded and decrypted IP traffic on machine M3 was 0.321 Mpps which was 10.8 % of generated IP traffic.

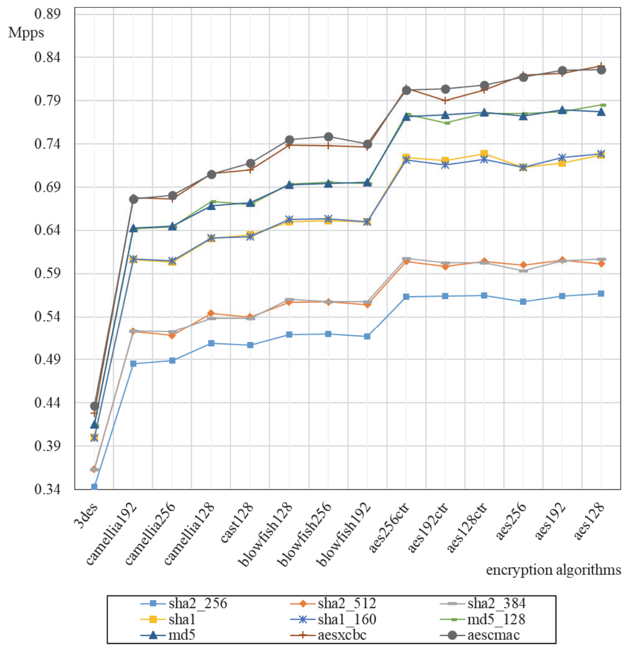

Fig. 9. Packet throughputs for separate encryption and integrity algorithms in hardware configuration B.

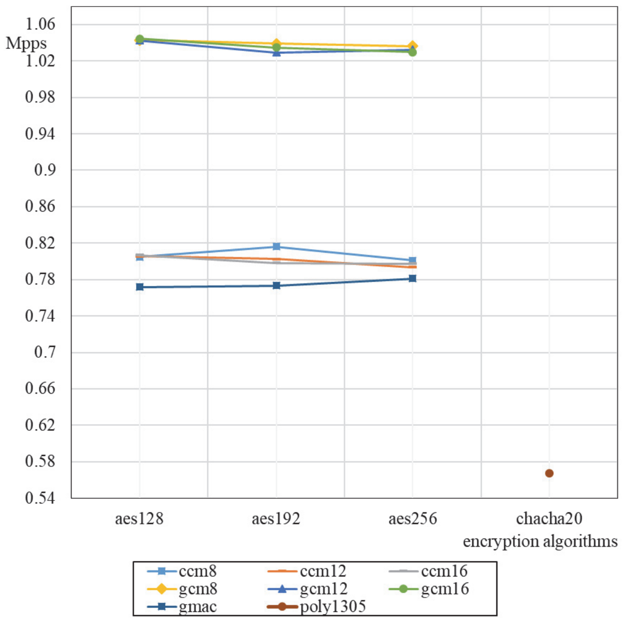

Fig. 10. Packet throughputs for the integrated encryption and integrity algorithms in hardware configuration B.

Fig. 9 shows the performance results for separate encryption and integrity algorithms in the case of hardware configuration B. As mentioned, in this case, the latest strongSwan distribution was used with the configuration that enables all supported encryption and integrity protection algorithms. The tests include all possible combinations of encryption algorithms, integrity algorithms and keys lengths. The results are similar as in the case of hardware configuration A. In this set of tests, the fastest encryption algorithm is again AES. Different key lengths of encryption algorithms did not have noticeable influence on performance. Also, in this set of tests, for configuration aes256-md5, the hardware configuration B provided 3.04 times higher packet maximal throughput compared to hardware configuration A, due to the more advanced CPU. Fig. 10 shows results for the second set of tests in the cases hardware configuration B. The fastest overall configuration for both sets of tests was aes128-gcm16 which achieved a throughput of 1.04 Mpps that was 3.4 % of the total generated IP traffic.

Finally, we tested all supported combinations of encryption and integrity protection algorithms with different IP packets lengths. As mentioned, IP packet length affects the maximal packet throughput – as IP packet length increases, the maximal packet throughput decreases.

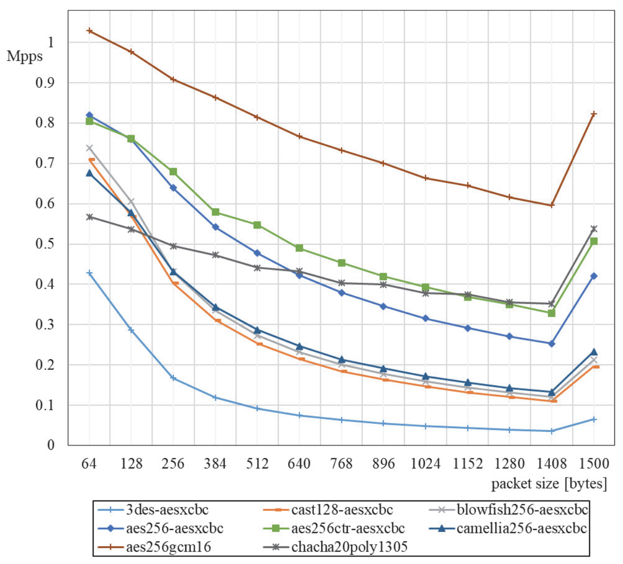

Fig. 11. Packet throughput for different strongSwan configurations on high-end commodity PC.

This test allows us to examine and analyze how packet length influences the performance. The behavior of all combinations of encryption algorithms and integrity algorithms is similar, and therefore for clarity, we used just one integrity algorithm (AES XCBC) and singled out results for the fastest configurations of each encryption algorithm. Also, we included results for the fast encryption algorithms with integrated integrity protection (AES GCM and ChaCha20 Poly 1305). Fig. 11 shows the packet throughput of different security configurations for various IP packet lengths. Usually, when software router forwards packets, the increase of packet length allows CPU more time to process each packet, because only the packet header is examined. In the case of encryption and integrity protection algorithms, CPU not only processes packet headers but also packet data. Thus, when packet length increases, CPU must process more data lowering further packet throughput. It is worth noting that as IP packet length reaches the maximum packet length of 1500 bytes, the packet throughput between machines M1 and M2 doubles. This is due to the packet encapsulation. As an IP packet is already at the maximal packet length, additional ESP header and trailer forces packet fragmentation. In this case, each ESP packet is divided into two packets which results in doubling packet throughputs. As Fig. 11 shows, almost all strongSwan configurations follow the similar decline curves of the packet throughput as the packet length increases, but ChaCha20 Poly1305 is an exception. The ChaCha20 is based on ARX (Addition-Rotation-XOR) instructions, which are CPU friendly instructions. Due to the different encryption protection scheme, ChaCha20 Poly1305 performance declines more slowly with the packet length increase compared to the other configurations. Thus, ChaCha20 Poly1305 performance exceeds almost all configurations except of the encryption algorithms that include integrity protection (AES GCM, AES CCM and AES GMAC).

VII. Conclusion

In this paper, we have analyzed functionalities and performance of software router with VPN features which utilized the Linux kernel. Integration of Quagga and strongSwan software platforms converts commodity hardware into the software VPN router. This configuration provides IPsec tunnels with higher resiliency to link failures. We compared performance of all encryption and hash algorithms supported by strongSwan and concluded that AES GCM provide the best performance.

References

[1] A. Sadasivarao, S. Syed and P. Pan, "Open Transport Switch - A Software Defined Networking," in *SIGCOMM*, Hong Kong, 2013.

[2] R. Bolla, and R. Bruschi, " Linux Software Router: Data Plane Optimization and Performance Evaluation," *Journal of Networks*, vol. 2, no. 3, 2007.

[3] P. Jakma and D. Lamparter, "Introduction to the Quagga Routing Suite," 2012. [Online]. Available: https://goo.gl/NXbOfL.

[4] "StrongSwan," secunet, 2016. [Online]. Available: https://www.strongswan.org/. [Accessed 1 10 2016].

[5] S. Padhiar and P. Verma, "A Survey on Performance Evaluation of VPN," *International Journal of Engineering Development and Research*, vol. 3, no. 4, pp. 516-519, 2015.

[6] S. Kent, "Security Architecture for the Internet Protocol," 2005. [Online]. Available: https://tools.ietf.org/html/rfc4301. [Accessed 1 10 2016].

[7] H. Redžović, A. Smiljanić and S. Gajin, "Performance Evaluation of Open-Source VPN Software Implementations," in *3rd International Conference on Electrical, Electronic and Computing Engineering*, Zlatibor, 2016.

[8] B. Schneier, "Description of a New Variable-Length Key, 64-Bit Block Cipher (Blowfish)," *Dr. Dobb's Journal,* vol. 19, no. 4, pp. 38-40, 1994.

[9] Intel, "Data Plane Development Kit," [Online]. Available: http://dpdk.org/.

[10] Intel, "Intel Advanced Encryption Standard (AES) New Instructions Set," 2016. [Online]. Available: https://goo.gl/UGyyfc. [Accessed 1 10 2016].

[11] H. Redžović, A. Smiljanić and B. Savić, "Performance evaluation of Software Routers with VPN features," *2016 24th Telecommunications Forum (TELFOR)*, Belgrade, 2016, pp. 1-4.