Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 32 (2014) 965 – 972

Procedia Computer Science

The 4th International Workshop on Frontiers in Ambient and Mobile Systems (FAMS)

Per-hop data encryption protocol for transmitting data securely over public networks

Rajitha Tennekoon[a]*, Janaka Wijekoon[b], Erwin Harahap[c], Hiroaki Nishi[d]

[a,b,c]*Hiroaki Nishi Laboratory,Department of Computer Science, Keio University, 3-14-1 Hiyoshi, Kohoku-ku, Yokohama, Kanagawa 223-8522, Japan*

[d]*Department of Systems Design, Keio University, 3-14-1 Hiyoshi, Kohoku-ku, Yokohama, Kanagawa 223-8522, Japan*

Abstract

It is a well-known fact that the Internet traffic travels through public networks. These networks lack security and are vulnerable. Encryption and public key cryptography are important technologies that are used to preserve data security and integrity, and to reduce information theft on the public networks. However, the existing routing protocols are incapable of providing secure data transmission on public networks. To this end, our laboratory introduced the Service-oriented Router (SoR) to maintain rich information for the next-generation networks by shifting the current Internet infrastructure to an information-based and an open-innovation platform. An SoR can analyze all packet stream transactions on its interfaces and store them in high throughput databases. Using the features of the SoR, in this paper, we propose a hop-by-hop routing protocol that provides per-hop data encryption. This infrastructure is proposing to preserve both the security and the privacy of data that traverses through public networks. We implemented a prototype of per-hop data encryption protocol on the ns-3 simulator and the results obtained are discussed in this paper.

Keywords: Service-oriented Router; data encryption; per-hop routing; ns-3

1. Introduction

Over the past few decades, several studies have been conducted regarding secure transmission of data over public networks [1-5]. On the other hand, the Internet is used by a majority of the worldwide population as their main communication media. It is a well-known fact that the Internet comprises of millions of public networks. In fact, public

* Corresponding author. Tel.: +81-45-566 -1785; fax: +81-45-566-1720.
E-mail address: rajitha@west.sd.keio.ac.jp

doi:10.1016/j.procs.2014.05.519

networks are less secure than private networks. Yet, it is used to transmit sensitive data belonging to millions of users across the world. The Internet's inherent problem of inadequate data security allows intruders to read and alter data streams.

Because sensitive data traverses through the Internet, people are demanding data security and privacy for their sensitive data. To this end, various security countermeasures have been invented and implemented in the current Internet infrastructure. A few major and well-known security protocols such as MACsec (IEEE 802.1AE) [1], Secure Sockets Layer (SSL) [2], Transport Layer Security (TLS) [3], Point-to-Point Tunneling Protocol (PPTP) [4], and Internet Protocol Security (IPSec) [5] have been implemented to provide data security. However, these methods can only provide end-to-end data security independent of the routing protocol and the user requirements.

Because the Internet comprises millions of networks that are connected by routers, most of the data travels through these network routers. We argue that the network router is one of the vital devices that can preserve data security and privacy during data transmission through the Internet. However, regular routers are incapable of providing adequate data security. To this end, the Service-oriented Router (SoR), which is a new generation backbone router, was proposed by our research team [7, 8]. An SoR has a high-throughput database. It stores squeezed data obtained from all transactions on its interfaces using a regular expression matching filter. In addition, SoRs can provide APIs for accessing stored contents in order to enrich services. Takagiwa et al. showed that the SoR is capable of handling 2 Gbps throughput [9], and that its hardware acceleration is capable of extending the processing throughput five times faster than the software-based acceleration [17]. In this study, we have used the functionalities of the SoR to implement a routing protocol that is capable of providing data security and privacy. We have used per-hop data encryption to maintain the data transmission consistency and data security. The implemented per-hop data encryption protocol consists of the following: (1) identifying neighbors 2) per hop key exchange (3) maintaining a key table in every SoR (4) providing link state routing (5) encrypting and decrypting packets on its arrival and departure to/from the SoR.

The rest of this paper is organized as follows: Section 2 discusses the background study and motivation. Section 3 discusses the development and operation of the per-hop encryption protocol, while Section 4 explains the simulation and test results. Section 5 presents the conclusion of the paper followed by future works.

2. Background study and motivation

Nowadays, the Internet has become the primary medium for communication. Therefore, people send their sensitive data through the public networks over the Internet. Owing to the lack of security in these networks, users and applications transmit their sensitive data through secured channels. MACsec (IEEE 802.1AE)[1], Secure Sockets Layer (SSL)[2], Transport Layer Security (TLS)[3], The Point-to-Point Tunneling Protocol (PPTP)[4] and Internet Protocol Security (IPSec)[5] are the main privacy preserving protocols used by the users and applications. Whenever data is encrypted or obscured by tunneling, it impedes analysis. This is applicable to SSL [2], IPsec [5], multicast, and various forms of non-secure tunneling [10]. In each of these methods, the key space remains constant throughout communication because it cannot be changed once it is set by the original sender. One of the main advantages of the proposed protocol is that the users can use dynamic key lengths among the hops, as per their preference. This allows a packet to be encrypted using different key spaces among the hops. The keys and their properties are managed by the SoRs and clients; otherwise, the applications can request their requirements regarding the key constraints from the SoR which are tailored to their needs. This facilitates encryption using the dynamic key spaces as required. Moreover, because the SoRs manage the keys, they act as an intermediate authorized router or gateway. An SoR can help in analyzing the packet contents, identifying and mitigating, or preventing threats such as intrusions and virus attacks; it can stop the breach earlier than the existing detection or prevention methods. Furthermore, it is difficult to install antivirus software in small ubiquitous devices that have limited processing performance, memory, and battery.

TCP/IP is the most commonly used Internet data transmission protocol and yet, is a heavy protocol because of the functions it was programed to provide. The Transmission Control Protocol (TCP) was introduced to provide a reliable connection-oriented communication channel between the two end hosts [11]. Although some critical applications require the reliability provided by the TCP/IP, occasionally they refuse to use the TCP/IP communication channels. This is because the multiple data buffering mechanisms place a heavy burden on these applications. Furthermore, these data buffering mechanisms lead to unpredictable delays. On the other hand, UDP/IP is widely used for time-critical applications as well as simple communication protocols such as routing protocols [12, 13]. Therefore, the proposed

routing protocol also uses the UDP/IP protocol as the underlying transmission protocol. The previous hop data retransmission method buffers the packets routed by an SoR and whenever an error is detected, the buffered packet can be retransmitted from the previous hop. This method helps in drastically reducing the data retransmission delay than the Forward Error Correction (FEC) method used in TCP. With the aid of the traceability function, a user or an application can request a trace back of the packet that they had received. This can then be used to verify the sender.

The proposed protocol performs a per-hop based encryption in the routing layer. In order to perform encryption in one SoR and decryption in another, both the SoRs must share a common symmetric key. The most difficult task in encryption is transmitting a secret key through the public networks. The proposed protocol uses the standard Diffie-Hellman key exchange algorithm in order to generate symmetric keys among neighbors by exchanging some values through the public network. The main advantage of this algorithm is that though the values are exchanged through the public networks, intruders who obtain the packets or sniff traffic may not be able to generate the shared secret.

2.1. Diffie-Hellman (DH) key exchange algorithm

The proposed protocol uses the well-known Diffie-Hellman key exchange algorithm to exchange a symmetric key between neighbors. The shared key can then be used to communicate securely through encryption. To initiate the DH key exchange, both the parties should agree on two non-secret numbers. The first number, denoted by ***g***, is the generator, and the second number, denoted by ***p***, is the modulus. These numbers can be transmitted through the public media, and the protocol uses two random prime numbers to denote them. For this process, the protocol uses the inbuilt ***rand()*** function to generate a random number and then verifies the primality of the generated number. After generating ***g*** and ***p***, both parties share the numbers between themselves through the connected link. In the next step, each party generates their private random secret value ***x***. Then, based on ***g***, ***p***, and ***x*** both the parties generate their public secret value Y using the following formula:

$$Y = g^{x} \bmod p \tag{1}$$

Using the above formula, the two parties will generate and exchange their shared secret value. Each party then exponentiates the received public value with the corresponding secret value to compute a common shared-secret value ***S*** using the following formula:

$$S = Y^{x} \bmod p \tag{2}$$

In the above formula, ***x*** refers to their respective private random secret value whereas *Y* and *p* are the shared values between themselves. At the end of this process, both the parties will generate the same shared secret *S*. Any attacker or sniffer listening on the channel will not be able to compute the secret value because even if the ***g***, ***p***, and the public secrets of both parties are known, at least one of the private random secret values is required to generate the common shared-secret value. Because only the communication parties know the private random secret values, it cannot be derived from any of the obtained values; the DH algorithm can create a shared secret value securely using the shared ***g***, ***p***, and ***Y*** values. Unless the attacker can compute the discrete algorithm of the above-mentioned equation to recover the ***x*** value of either ends, the attacker cannot obtain the shared secret value [14].

2.2. Per-hop encryption

There are two methods of encryption namely link encryption and end-to-end encryption. In the link encryption or online encryption, all data found along a link is encrypted regardless of its content or the protocol. In this method, the payload, headers, and trailer are encrypted as a whole. Therefore, the packets must be decrypted at each hop to enable the router or other intermediate device to know where to send the packet next. The router must decrypt the header portion of the packet, read the routing and address information contained in the header, and t hen re-encrypt the information and send it further. However, in end-to-end encryption, only the payload will be encrypted leaving the

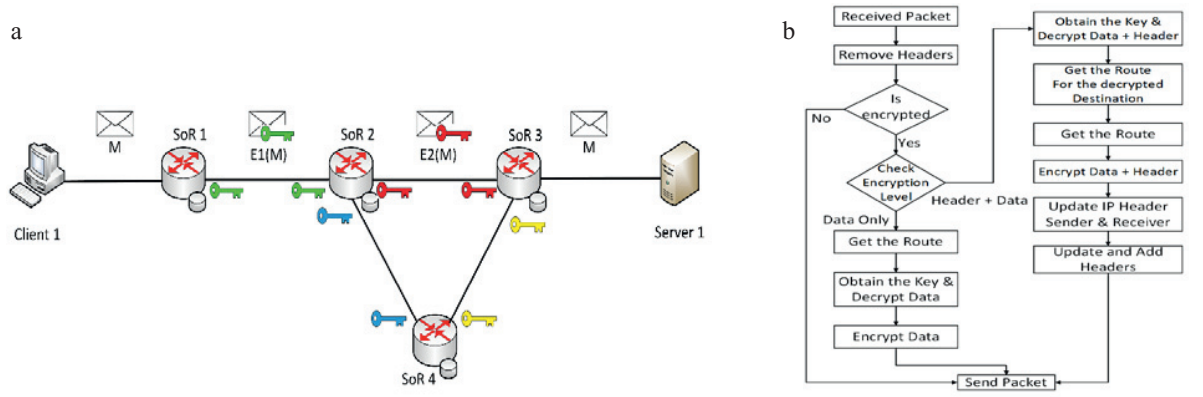

Fig. 1. (a) Per-hop data encryption. (b) Flow chart for packet encryption process.

headers and trailers readable. This leads to vulnerabilities in learning sensitive information such as the sender details, the destination of the packet, or the type of data it is carrying. In case of end-to-end encryption, decryption and encryption of packets at each hop is not required, because the headers and trailers are not encrypted. Therefore, attackers or a person who interprets the packet will be able to obtain this information easily. Often, in communication, we use end-to-end encryption in techniques such as SSH[6] and SSL[2]. Moreover, we use link encryption in techniques such as MACsec (IEEE 802.1ae) [1] in Ethernet and Point-to-Point Tunneling Protocol (PPTP)[4] mainly in satellite links and T3 lines, where we need to encrypt all the transferred data.

End-to-end encryption is usually initiated by the end-to-end hosts. In end-to-end encryption, once the encryption properties such as the key length and the encryption algorithm are decided, they cannot be changed throughout the communication. On the other hand, link encryption is used for performing encryption and decryption process between two neighboring devices. In such a scenario, all nodes between the end hosts must either encrypt or decrypt packets or perform both encryption and decryption. In fact, end-to-end encryption happens within the applications, and the link encryption occurs at the data link and physical layers.

Unlike in end-to-end encryption, in the proposed per-hop encryption protocol, packets are encrypted while they are being propagated through the network. As illustrated in Fig. 1(a), the packets will be encrypted by the SoRs as they propagate through the SoR network. Initially, SoRs securely exchanges keys respective to their connected links using the DH key exchange algorithm and stores them in a key table. Then, when SoR#1 receives a packet from the client, it retrieves the next hop based on the destination IP address obtained from the routing protocol. Next, it retrieves the shared key between SoR#1 and SoR#2 from the key table and encrypts the packet using the key, and sends it to SoR#2. Once the packet arrives at SoR#2, then SoR#2 decrypts the packet using the shared key between SoR#1 and SoR#2. In the next step, SoR#2 determines the next forwarding hop based on the packet destination and encrypts and forwards the packet to the next hop. This scenario continues until the packet reaches its destination.

Moreover, the proposed protocol will be able to provide flexibility in selecting the encryption level of the packet in two ways namely: encrypting only the packet payload, and encrypting both the packet payload and the headers. This process is illustrate in the Fig. 1(b) flowchart. If the packet requires encryption of both the packet headers and the payload, then the SoR will first obtain the original sender's and receiver's IP addresses of the packet from the original IP header. Then, these data will be written on the SoR packet header along with the payload used by the protocol. In the next step, the entire SoR packet containing the payload and the SoR header will be encrypted. Thereafter, the sender and the receiver portions of the original IP header will be replaced with the current SoR forwarding interface IP and the IP obtained for the next hop SoR respectively. After that, the altered IP header will be attached to the new packet. This helps in routing the packet on the optimum path through the existing intermediate network devices such as standard routers or layer 3 switches as the packet contains an IP header. Henceforth, the content along with the original sender and receiver data will be unreadable by any device other than the destined next hop SoR. The destined next hop SoR is the only device that has the key to decrypt the packet, obtain the original data, the originated source, and the final destination. If the user or the application requires only encrypting data, then the payload of the receiving

packet will be encrypted without altering or copying the original IP header. Furthermore, because the proposed protocol encrypts data in the hop-by-hop manner, the encryption key length and the encryption algorithm can vary between any two hops. This provides more flexibility to the users, applications, and the network administrators.

3. Proposed per-hop encryption protocol development and its operation

The proposed protocol is implemented in the well-known simulator, Network Simulator 3 (ns-3). ns-3 is the successor of Network Simulator 2 (ns-2) [15]. ns-3 is fully modularized according to the real world, and though it is a simulator, it provides the architecture and environment that is more realistic than any other network simulator currently available [16].

SoRs create their neighbor tables using the Hello and Hello Reply packets exchange process. Upon trigger, the SoR interfaces up method and exchanges Hello packets with its neighbors. After receiving a Hello packet, the SoR responds to it with a Hello Reply. This process is essential because it enables the protocol to identify the links as bidirectional links. After exchanging the Hello packets, SoRs initializes the DH shared key distribution process with every neighbor in the neighbor table. Upon successful key exchange, the SoRs generates a key table and adds key information, neighbor information, and its local interface information. The protocol is flexible enough to renew the keys at the user's or application's request. This is achieved via retriggering the DH algorithm between any SoRs accordingly. Further, the underneath IPv4 routing protocol exchanges the routing information simultaneous to the key exchange process. Therefore, the proposed per-hop data encryption protocol is capable of exchanging both routing and DH key information between neighbors with minimum delay. Consequently, the end users are not required to perform key exchange prior to the connection initiation that is an essential requirement of the end-to-end encryption technologies. The topology inherently does this automatically.

Whenever an SoR receives a packet or a data stream that needs to be routed, it first checks whether the packet or stream requires encryption. This capability enables the proposed protocol to act as an intermediate between the existing link encryption and end-to-end encryption methods as it allows both unencrypted packets and encrypted packets to be routed simultaneously. Moreover, it will enable the SoR to reduce its burden in encrypting and decrypting all the packets that flow, and to facilitate both encrypted and unencrypted communications simultaneously according to the client or application requirements. On the one hand, if the received packet requires the encryption service, then, the packets will be routed after being encrypted by the SoR. On the other hand, if the SoR receives an ordinary packet, which does not require encryption, then it will treat the packet as a standard packet and route it as an ordinary link state routing protocol.

After receiving an encryption-required packet, SoR will first check its routing table and select the most efficient next hop required to forward the packet. Then, the SoR will obtain the particularly shared key according to its key table. Next, it encrypts the packet content using the obtained key. Moreover, the encryption level can be selected according to the requirements of the client or the application.

In a communication where only the data needs to be encrypted without any other information, the proposed protocol will only encrypt the payload and send it to the next hop SoR along with the original IP header and other SoR header information. This permits successful identification of the packet at the next hop SoR. The SoR header will include fields that uniquely distinguish each packet type without any conflicts. The SoR header used in this method will be explained in the next section. Upon successful retrieval, the next hop SoR, which receives the packet, will decrypt the packet using the same shared key obtained from its key table. Then, after reading the original sender and receiver details, if it is not addressed to itself, the above process is repeated after selecting the most effective and efficient neighboring hop to forward the data. Further, if the packet requires encryption, it will be encrypted using the same process, and the packet will be forwarded to the appropriate next hop SoR. This process continues until the final SoR sends the packets to the receiver. At the destination, if the packet is completely encrypted, the original sender and receiver details will be reversed to the IP header, so that the receiver will receive an identical packet that was sent by the sender. The main advantage of this method is that the most efficient and effective path can be selected according to the real time link conditions. This will speed up the packet transmission while optimizing the congestions in the wide area networks. In addition, the protocol will allow simultaneous transmission of both ordinary packets of the network and the packets that need to be securely delivered via encryption. Meanwhile, the encryption level and the

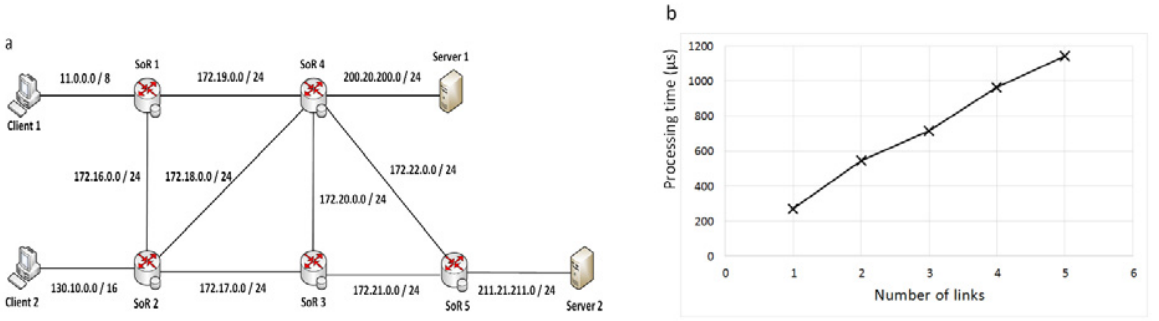

Fig. 2. (a) Simulation topology. (b) Consumed processing time for DH key exchange in SoR#4.

key size will also be configured dynamically according to the sensitivity of the data or the client and the application requirements.

In this implementation process, a router or an SoR can use different individual key lengths for all their interfaces. Nowadays, in case of encryption, many nations have imposed restrictions on key lengths for the data travelling through their networks. The proposed method helps in overcoming this by maintaining the required key length based on these requirements in every interface as necessary. Therefore, whenever the rules change, the key lengths can be managed centrally thereby giving total control to the nations according to their individual requirements.

4. Simulation, experiment and test results

The simulation is executed on a machine with a CPU of Intel Core i7 3.2 GHz, 24 GB RAM, 2 TB hard disk space and running CentOS 6.5 64-bit operating system. The proposed routing protocol is implemented in the simulation software, Network Simulator 3 (ns-3). The main drawback of the proposed protocol is the delays caused by the encryption and decryption processes while routing encrypted packets. In the implementation, Advanced Encryption Standard (AES) is used as the encryption algorithm with a 256-bit key length. Further, there tends to be an initial time cost until the Diffie-Hellman algorithm is completed. Therefore, the key table creation time has been measured in a network topology, as illustrated in Fig. 2(a). All the links in the topology were configured with a throughput of 5 Mbps and a 2 ms delay. While obtaining the test results, authors have obtained the total processing time consumed by the processor in executing the particular functions in microseconds using the real time stamp counter of the CPU.

According to Fig. 3, the busiest SoR in the network is SoR#4 because it is connected to five links resulting in managing a maximum number of links. Therefore, the processing time taken by SoR#4 in generating DH keys while increasing the links from one to five were individually measured, and the results were plotted as illustrated in Fig. 2(b). From the results, we can substantiate that the processing time taken for the DH key generation process increases with an increase in the number of links attached to an SoR. When the SoR contained only a single link, the processing time was 268.33 µs; this increased to 1140.48 µs when the SoR was attached to five links. These processing times also include the processing time consumed by the route propagation of the protocol that runs simultaneously with the key exchange.

Furthermore, we measured the time taken for the decryption and encryption processes of the proposed protocol. The test was executed using the topology given in Fig. 2(a). In this test case, Client#2 was programmed to send the encryption required packets destined to the Server#1 as presented in Fig. 2(a). Client#2 was started 0.5 s after the start of the simulation. Then, Client#2 sent a fixed number of packets at an interval of 0.02 ms. The first 0.5 s were allocated to create and exchange the DH keys, create key tables, and propagate the routing tables of the proposed protocol. These processes were executed automatically by the protocol regardless of the network topology and the number of SoRs in any topology. The packets travelled through the path SoR#2, SoR#4 and SoR#5 to reach Server#2. This path was selected automatically by the protocol since it was the least cost path to reach the destination. Moreover, the main advantage of using this path is that the packet would travel through SoRs that contained connected links such as four, five, and three accordingly. Therefore, this resulted in measuring the processing times of SoR#2, SoR#4, and SoR#5

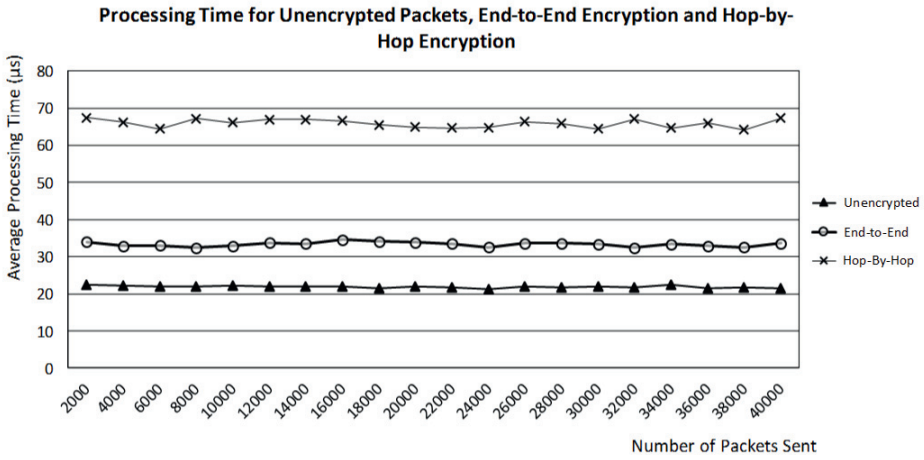

Fig. 3. Processing time taken for unencrypted, end-to-end encryption and hop-by-hop encryption packet transmission.

where the burden of the SoRs increased according to the number of connected links.

We measured the processing time in SoRs: SoR#2, SoR#4 and SoR#5, for following scenarios. 1) Process unencrypted packets, 2) process end-to-end encrypted packets and 3) process hop-by-hop encrypted packets. Further, we measured the processing time for 20 test cases using 2,000 to 40,000 packets and we used average processing time for the comparison. The results comparison is given in the Fig. 3. According to the Fig .3, SoRs consumed averagely 21.85 µs to process unencrypted packets. Furthermore, end-to-end encryption packet processing consumed averagely 33.31 µs, which is 34.41% increment. That was expected on the SoR due to the extra process consumption for manage key tables and process the end-to-end encryption packets. Finally, SoR consumed averagely 65.86 µs for the processing of hop-by-hop encryption packets. That is due to manage key tables , routing and, further SoRs have to decrypt and encrypt every packet in each hop. Nonetheless, it is average 66.82% increment compared to unencrypted packet transmission, and 49.42% increment compared to end-to-end encrypted packet transmission. However, as each and every SoR pair use an unique key, the original content of the packet can be obtained via decryption only by the destined SoR. Therefore, compared to other two methods, the proposing method shows much secure data transmission over public networks.

Finally, we measured the end-to-end delay for all aforementioned three cases. The results are plotted in the Fig. 4. According to the Fig. 4, an unencrypted packet consumed about 4.0219 ms to propagate from Client#2 to Server#1. Nevertheless, end-to-encrypted packet consumed about 4.0333 ms to propagate from Client#2 to Server#1.

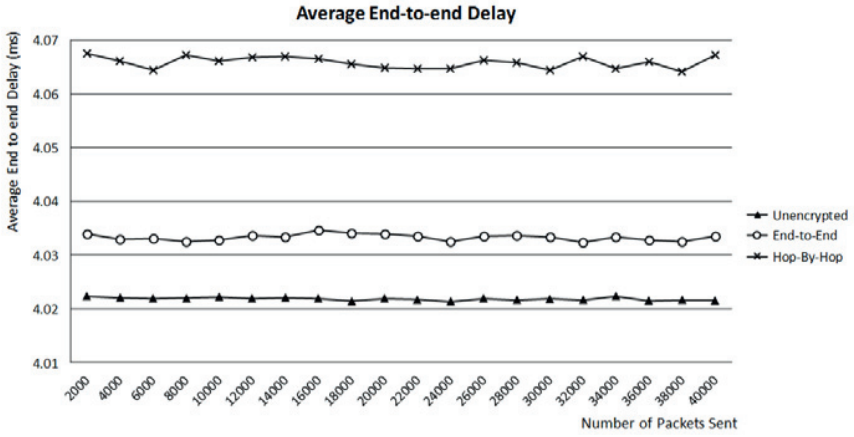

Fig.4. Average end-to-end delay.

According to the Fig. 4, the proposing method consumed about 4.0659 ms and it is 0.08% amount increment compared to end-to-end encryption. That is because SoRs consumed additional processing time for the decryption and encryption processes in each hop. Even though, the proposed method displays 0.03 ms increment of packet propagation delay, it is clear that the proposing method has the merit as the most concerning point over public networks as security.

5. Conclusion

In this paper, we introduced a per-hop data encryption protocol that can be used to transmit sensitive data over public networks. The proposed data encryption protocol can simultaneously provide both secured routing and ordinary routing according to the user requirement. Moreover, the test results showed that the proposed method can be used with end-to-end encryption as well. Furthermore, the process method is able to provide higher security to the data travers through public network by ensuring integrity and confidentiality. However, those flexibilities come with the cost of additional 0.044 ms end-to-end delay in data transfer.

In future works, we will test this method in a real world network with real traffic. Moreover, as mentioned earlier in section 2, previous hop FEC method will be implemented. In addition, for further security, we will implement the Certification Authority (CA)-based authentication architecture to authenticate SoRs, clients, and applications.

Acknowledgements

This work was partially supported by Funds for integrated promotion of social system reform and research and development, MEXT, Japan, by Low Carbon Technology Research and Development Program for "Practical Study on Energy Management to Reduce CO2 emissions from University Campuses" from Ministry of the Environment, Japan and by MEXT/JSPS KAKENHI Grant (B) Number 25280033.

References

1. IEEE Standards Association (2011), Media Access Control (MAC) Security. [Online]. Available: http://standards.ieee.org/getieee802/download/802.1AEbn-2011.pdf.
2. Freier, A., Karlton, P., and Kocher, P., “The Secure Sockets Layer (SSL) Protocol Version 3.0”, IETF RFC 6101, August 2011; http://tools.ietf.org/html/rfc6101.
3. Dierks, T., and Dierks, T., “Transport Layer Security (TLS) Protocol Version 1.2”, IETF RFC 5246, August 2008; http://tools.ietf.org/html/rfc5246.
4. Hamzeh, K., et al., “Point-to-Point Tunneling Protocol (PPTP)”, IETF RFC 2637, July 1999; http://tools.ietf.org/rfc/rfc2637.txt.
5. Frankel, S. and Krishnan, S., “IP Security (IPsec) and Internet Key Exchange (IKE) Document Roadmap”, IETF RFC 6071, February 2011; http://tools.ietf.org/html/rfc6071.
6. Ylonen, T. and Lonvick, C., “The Secure Shell (SSH) Transport Layer Protocol”, IETF RFC 4253, January 2006; http://tools.ietf.org/html/rfc4253.
7. K. Inoue, D. Akashi, M. Koibuchi, H. Kawashima and H. Nishi, "Semantic router using data stream to enrich services", 3rd International Conference on Future Internet Technologies (CFI08), Seoul, Korea, June-2008.
8. Janaka Wijekoon, Erwin Harahap, Hiroaki Nishi, Service-oriented Router Simulation Module Implementation in NS2 Simulator, Procedia Computer Science, Volume 19, 2013, Pages 478-485, ISSN 1877-0509.
9. Takagiwa, Kenichi; Ishida, Shinichi; Nishi, Hiroaki, "SoR-Based Programmable Network for Future Software-Defined Network," Computer Software and Applications Conference (COMPSAC), 2013 IEEE 37th Annual, vol., no., pp.165,166, 22-26 July 2013.
10. Harris, S., CISSP All-in-One Exam Guide, McGraw-Hill Companies, 2008.
11. DARPA Internet program. “Transmission Control Protocol”, IETF RFC 793, September 1981; https://www.ietf.org/rfc/rfc793.txt.
12. Moy J., “OSPF Version 2”, IETF RFC 2328, April 1998; http://www.ietf.org/rfc/rfc2328.txt.
13. Malkin G., “RIP Version 2”, IETF RFC, November 1998; http://tools.ietf.org/html/rfc2453.
14. Diffie, W., Hellman, M.E., "New directions in cryptography," Information Theory, IEEE Transactions on , vol.22, no.6, pp.644,654, Nov 1976.
15. Chung J., Claypool M., (2011) NS by Example. [Online]. Available: http://en.wikipedia.org/wiki/NS-2.
16. Abraham J. (2013), ns-3 manual. [Online]. Available: http://www.nsnam.org/docs/manual/singlehtml/index.html.
17. Yusuke Nishida, and Hiroaki Nishi, Implementation of a Hardware Architecture to Support High-speed Database Insertion on the Internet, The 2012 International Conference on Engineering of Reconfigurable Systems and Algorithms (ERSA2012 in WORLDCOMP2012), Online proceedings six pages, 2012.