Indonesian Journal of Electrical Engineering and Computer Science
Vol. 22, No. 1, April 2021, pp. 442~449
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v22.i1.pp442-449

Embedding the three pass protocol messages into transmission control protocol header

Suherman[1], Deddy Dikmawanto[2], Syafruddin Hasan[3], Marwan Al-Akaidi[4]
[1,2,3]Electrical Engineering Department, Universitas Sumatera Utara, Medan, Indonesia
[4]American University in the Emirates, Dubai, UAE

Article Info

Article history:

Received Dec 29, 2020
Revised Feb 23, 2021
Accepted Mar 3, 2021

Keywords:

Key sharing
Network security
Three pass protocol
Transmission control protocol

ABSTRACT

Transmission control protocol provides reliable communication between two or more parties. Each transmitted packet is acknowledged to make sure successful deliveries. Transport layer security protocols send security information exchange as transmission control protocol (TCP) loads. As results, the handshaking stage experiences longer delay as TCP acknowledgement process has already been delay prone. Furthermore, the security message transfers may have their own risks as they are not well protected yet. This paper proposes TCP-embedded three pass protocol for dynamic key exchange. The key exchange is embedded into TCP headers so that transmission delay is reduced, and message transfer is secured. The proposed protocol was assessed on self network by using socket programming in lossless environment. The assessments showed that the proposed protocol reduced three-pass protocol message transfer delay up to 25.8% on lossless channel. The assessment on security also showed that TCP-embedded three pass protocol successfully secured each transmitted TCP load using a unique key; that is much securer than the compared method.

Corresponding Author:

Suherman
Electrical Engineering Department
Universitas Sumatera Utara
Medan, Indonesia
Email: suherman@usu.ac.id

1. INTRODUCTION

Packet switching based networks slightly change in the last decades as transmission control protocol (TCP)/IP protocol and its applications grow rapidly, especially on mobile network [1]. As results, social media applications dominate the internet traffics, including multimedia contents for both social and business applications. The increasing demands on multimedia content exerts real-time and security issues. Real-time issues relate to the requirement of low latency message, audio or video transfer. Security concerns with the message ownerships and message contents. Low latency requirement of multimedia packets has been fulfilled by lightweigh coding [2] and fast end to end transfer protocol [3]. Security matters deal with encryption, authentication and authorization [4].

This paper concerns with shared-key delay and security requirement. Delay in end to end protocols has been considered when designing multimedia communication application either by using unreliable user datagram protocol (UDP), reliable transmission control protocol (TCP) or combined protocols [5]. Compact audio video codings have contributed on the more TCP implementation rather than UDP. Even dough, TCP process should be minimized as efficient as possible as frequent acknowledgment increases latency.

On the other hand, security issues have risen as mobile networks that contributed on traffics and user increments have openly exposed physical infrastructures. More attacks and treats exist in wireless link. Djenouri [6] explained that more risks are due to multi-hop transmissions, node movements, network amorphous, power limitations and computation strength on user equipments. Emerging network paradigms such as cloud and fog also introduce more security issues [7-11].

One of the biggest fields of security subjects is the encryption as it is powerful enough to protect the information content from unintended users anywhere in networks. However, shared-key exchange to enable encryption and decryption generates additional real-time issue: latency. Moreover, those messages are transported by TCP and shared by using key exchange protocol such as three pass protocol (TPP). TPP resolves key exchange by using three stages of message transfer. TPP has been used to enable internet secure transport protocol [12], to encrypt heterogeneous network messages with shared keys [13], to secure data flow in IoT networks [14] and to protect RFID networks [15]. TPP processes involve message exchanges between transmitter and receiver. The encryption password or shared key is transported by using three messages between transmitter and receiver. Each transmitter or receiver performs encryption decryption by using default symmetric algorithm such as one-time pad protocol (OTP). Initially, the encryption password is treated as a text, encrypted by using key A in sender. The cypher text is then sent to other receiver who encrypts this cypher text by using key B. Receiver then transmitts it back to Sender. Sender decrypts the message to produce cypher text B and sends it back to receiver. Receiver decrypts this message and finds the encryption key. The transmitter and receiver encrypt and decrypt message once.

OTP is used to encrypt the messages. Problem increases when the third party obtains all three messages that enables the password to be cracked by using simple exclusive addition. OTP in TPP has been modified by using various encryption techniques such as H-Rabin [16], Hill Cipher [17], Viginere Cipher [18] and El-Gamal [19]. TPP modification can also be made for single or multiple links of TPP message transfer [20]. Similar security protocols are also available for transport layer such as QUIC [21] and transport layer security (DTLS) [22]. DTLS is an enhancement of TLS [23]. All these security protocols added additional public key in their implementations which inject additional delay. This delay increment potentially decreases the end to end performances.

Communication processes between transmitter and receiver are performed end to end. The messages are transported by the transport protocol either TCP or UDP, depending upon the application designer. TCP as a reliable protocol provides connection-oriented services so that the transmitted packets are acknowledged once they reached destination. Next packet from different window will not be transmitted before the acknowledgement of previous packet is received by sender. As result, transmission may take longer delay. Packet transmission is ensured by TCP, but packet security depends on the encryption technique.

On the other hand, UDP provides fast transmission services without acknowledgement. Every packet is sent as soon as possible to reach destination. Transmission link ignores the probable events such as network congestion or receiver overflow. As results, congestion in the link and receiver may get worsen. Moreover, any packet loss occurred will not be corrected. TCP and UDP have been modified in many ways to further improve the performances [24]. However, TCP dominates the implementation as transmission link speed increases significantly nowadays, and information codings has been effectively delivering the multimedia traffics. Figure 1 (a) shows TCP stages, TCP starts the communication process by using the three-way handshake (SYN, SYN-ACK and ACK) [25]. TCP ensures all transmitted packets be acknowledged (ACK). TCP also terminates all communication when ended by using FIN and ACK. This means, TCP guarantees the communication conducted safely between two parties from the beginning to the end.

However, in term of packet security, TCP relies on the encryption techniques provided by the upper layer. This means encryption techniques are part of the application layer within the TCP/IP architecture. The existing security protocol on transport layer such as socket security layer (SSL) also performs key exchange. SSL is developed to enable the web browser and web server to communicate securely by authenticating the server and client. SSL uses a public key to encrypt the transferred data. Beside SSL, transport layer security (TLS) also works in the same layer. TLS is a cryptographic protocol working in between the transport and the application layers. It supports various services like social networks, webmail, video streaming, including the HTTP Secure application [26]. As shown in Figure 1 (b), TLS imposes a long process. The first stages are to handshake and exchange both parties' information including the encryption techniques. The next process is to make sure the shared key is exchanged properly. After these long processes, the encrypted-application-data is then transmitted. Other security solutions in transport layer application is performed within transport layer by ensuring the integrity of bits within TCP packet. This method may use full or partial checksum. Protocols like TCP-Lite [27] and UDP-Lite [28] ensure bits or data integrity form error or attack by using specific checksum bits.

The most potential problem occurred on shared-key encryption method is when key transmission is failed; then the transmitted message cannot be translated properly by user [29]. This phenomenon often occurs when receiving messages in social media such as whatsapp, where messages are failed to display. This problem also harms the data collection within the IoT application, where important data may not be available to be harnessed.

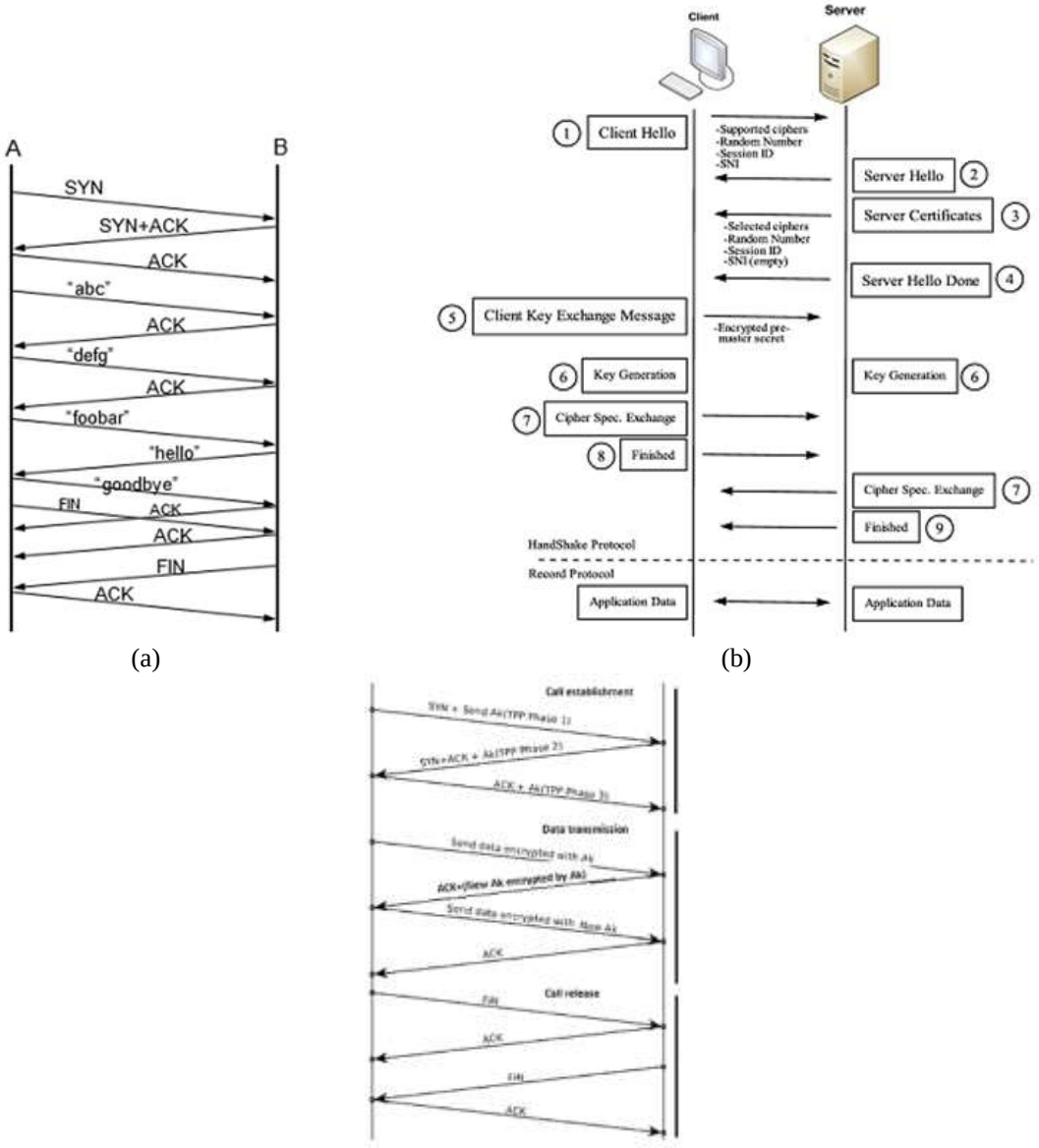

Figure 1. The message transfer: (a) TCP, (b) TLS, and (c) the proposed protocol

On the other hand, real-time application requires data transaction processing as fast as possible. Aforementioned solutions such as TLS and SSL require long time to process shared keys. Furthermore, the key is used for all sessions of data transfer that minimize the security of messages. This paper proposes TCP-embedded secure message transfer so that delay on shared key exchange is minimized and key is dynamically exchanged easily.

This article is arranged as follows. The proposed method on how shared key is inserted to TCP header is described in following proposed method session. The evaluation methods by using a self network with java socket programming for both server and client are explained afterwards. Results on evaluation of lossless environment are discussed in evaluation results session. Conclusion of the research finding is given at final section.

2. PROPOSED METHOD

This paper proposes the key exchange technique that makes use the packet safety and reliability provided by TCP. TPP messages are embedded into TCP headers so that the handshaking and key-sharing processes can be united. The proposed method also embeds the dynamic keys into the header of data acknowledgement (ACK) packets so that the encryption keys change dynamically for every TCP window, so the packets sent in different windows are encrypted using different keys. The proposed method is depicted in Figure 1 (c). During the TCP call establishment process, the initial active key sharing on TPP is also established. The three TPP messages are embedded into the three TCP handshaking messages: SYN, SYN-ACK and ACK packets. At the end of the TCP call establishment process, the active key Ak is expected to have been also delivered. So, there is no necessary to have other key exchange process. Compared to TLS as depicted in Figure 1 (b), the proposed method is much simpler. The proposed method minimizes the number of transmitted packets and reduces the processing time.

Data transmission is performed after data is encrypted in sender by using the shared key. Once data arrives at the receiver, TCP will send the ACK packet. Before ACK transmission is performed, receiver generates another key (new Ak) for next window encryption in sender. This new key is encrypted by using previous key and the cyphertext is inserted into ACK header. The potential additional delay in the new shared-key generation can be anticipated by adding slightly the ACK timer. After receiving ACK packet, sender extracts cyphertext from ACK header, decrypts it and uses the key for encrypting the next packets. These processes may also add delay. If the initial encryption complexity is O(1), the proposed method will increase it to O(n), where n is the number of packets or the TCP windows size.

3. EVALUATION METHOD

In order flexibly evaluate the proposed method, UDP datagrams is utilized to model TCP process. The code implementation was in java socket programming language. The code snippets are shown in Figure 2. The code is valid only for TCP window size 1. It starts by initializing UDP datagram for all TCP messages. The first message is TCP SYN preceded by TPP key generation. The generated key is inserted to TCP SYN message. The next step is to receive SYN+ACK+TPP phase 2 message. TPP phase 3 is then processed and sent with TCP ACK. Data encryption, data transmission and ACK+newKey process are repeated 10,000 times. Finally, the call release codes conclude the sender program. The receiver program processes TPP messages, new key generations and data decryption.

```
//import library
public class Sender{
    //codes for global variables
    public static void main(Strine[] args){
        //codes for network exceptions
        //running servers
        accessServer(); }
    private static void accessServer(){
        //codes for local variables
        try{
            //code for datagram creation
            //codes TPP encryption key generation
            try{
            //codes for SYN+TPP phase 1
            //codes for Fitt time record
            start = System.currentTimeMillis();
            outPacket = new DatagramPacket((messageA+ 1\n").getBytes(),
            (messageA+ "0").length(),host,PORT);
            datagramSocket.send(outPacket);
            //codes for SYN+ACK
            //Receive ACK+TPP phase 2
            buffer = new byte[256];
            inPacket = new DatagramPacket(buffer,buffer.length);
            datagramSocket.receive(inPacket);
            //Codes for TPP processing
            //Codes for ACK+TPP phase 3
            outPacket = new DatagramPacket((messageA+ "01").getBytes(),
            (messageA+ "01").length(),host,PORT);
            datagramSocket.send(outPacket);
            //Record TPP time stop = System.currentTimeMillis(); }
            catch(Exception e){
                //codes for exceptions }
            do{
                startEnc=System.currentTimeMillis();
                //codes for encryptions
                stopEnc=System. currentTimeMillis();
                startTrans=System.currentTimeMillis();
                //codes for data transmission
                //codes for ACK and new key revealations
                stopTrans=System.currentTimeMillis(); }
            while(i<10000); }
        catch(IOException ioEx){ ioEx.printStackTrace(); }
        //codes for call termination }
```

(a)

```
//import library
public class Receiver{
    //Codes for global variables
    public static void main(Strine[] args){
        //Codes for exceptions
        handleClient(); }
    private static void handleClient(){
        //code for local variables
        try{
            //codes for datagram
            do {
                //receiving datagram
                messageIn = new String(inPacket.getData(),
                0,inPacket.getLength());
                if(messageIn.equals("SYN")){
                    //codes for processing TPP1
                    //codes for sending SYNA-ACK+IPP phase 1
                }else if(messageIn.equals("ACK+TPP3")){
                    //codes for processing TPP3
                    //codes for processing key
                }else if(messageIn.equals("DATA")){
                    //codes for decrypting data
                    //codes for processing new key
                    //codes for processing ACK
                }(messageIn.equals("RELEASE")){
                //codes for processing call release }
            }
            while(! connected);
                //Check parameters
                System.out.println("Average dec: "i-allDecii." sec");
                System.out.println("Packet loss: "+loss/i);
        }catch(IOException ioEx){
            ioEx.printStackTrace();
        }
        finally{
            //close socket
            System.out.println( \n Closing connection... ");
            datagramSocket.close(); }}
```

(b)

Figure 2. Code snippets for evaluation; (a) sender code and (b) receiver code

Assessment is conducted by using lossless self-network of IP 127.0.0.1, by using a computer with Intel(R) Core(TM) i5-2430M CPU @ 2.40GHz processor, memory of 4 GB, 640 GB hard disk, and Netbeans java editor. Sender sends 10,000 packets to receiver. Assessment is repeated 50 times. Assessment of the proposed method is mostly to determine the processing delay for the key transfer, key generation, message encryption and message decryption. The proposed method results are compared to the separated key-exchange schema such as in TLS. Since this paper focuses on the capability of TCP header in carrying TPP messages, lossy channel is not yet tested and out of scope of this paper.

4. EVALUATION RESULTS

Assessment on the local lossless network shows the low round trip time for all TCP connections. However, the time is distinguishable for both existing and proposed methods. Figure 3 (a) shows comparison of TCP data transfer without encryption and with encryption with key shared by using the proposed method. The round-trip-time of the proposed method increases slightly about 0.6%. This increment is not significant as encryption process within the proposed method incurs only about 0.16 ms additional delay. This delay increment comes mostly from decryption process to reveal the key from the TPP process.

As shown in Figure 3 (b), key revelation contributed almost 0.4 ms in average, higher than key generation process that exerted only about 0.13 ms in average. The remining delay was additional transmission time. On receiver, after receiving packets of one window, receiver should generate a new key for the next packet transmission. This key is encrypted by using previous key and sent within the ACK packet. In sender, the received ACK will be extracted to find new key cyphertext and decrypted using previous key. Both processes in sender and receiver contribute to roud trip time increment. In the real system, this will not affect the performances as processor speed and memory resources compensate them.

Compared to existing TPP, the proposed method combines call establishment and TPP stages. These two processes have different processing time. Figure 3 (c) shows the TPP and TCP call establishment delay comparison. The average time for TPP stage is about 15.5 ms, while call establishment requires about 46 ms. In traditional method, both processes are executed separatedly and sequentially that makes up more than 60 ms delay. The key transmission in existing solution is performed by two separated processes: TCP call set up and TPP messages, which makes the total time for key exchange is higher than in the proposed protocol. Figure 3 (d) shows TCP comparison when TPP and call establishment are performed separately or at once as in the proposed method. The transmission time reduction by the proposed method reached up to 24.8 % in average. TPP and TCP call establishment delays in Figure 3 (c) is replaced only by TCP call establishment in the proposed method. This is equivalent to 15.5 ms delay reduction. Meanwhile, the encryption and decryption processes within the proposed method produce higher encryption time and lower decryption time as shown in Figures 3 (e) and 3 (f). However, this value is depending on the text length and the encryption algorithm as in Figure 3 (g). Text length and algorithm determine the overall delay. OTP has the lowest delay and AES the highest delay.

In term of data security, when security attacks occur and the key is successfully extracted, all packets transported by TCP that employs the traditional TPP will be compromised as dynamic key is not employed. When dynamic key is employed, it can protect the other 9,999 packets. However, delay increases significantly as each key requires its own TPP process. On the other hand, for the proposed method, one compromised key is only affecting one packet as every packet has unique key. The dynamic key is easily applied in ACK packet. The security comparison is outlined in Table 1.

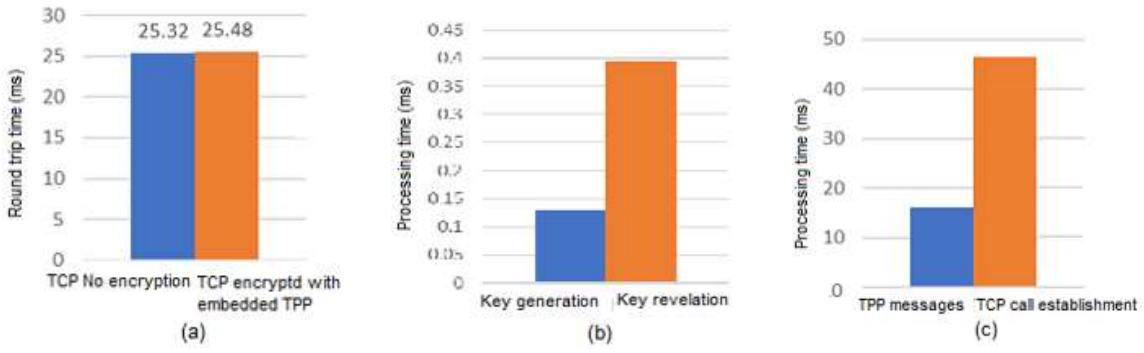

Figure 3. Round-trip-time and the increment reasons

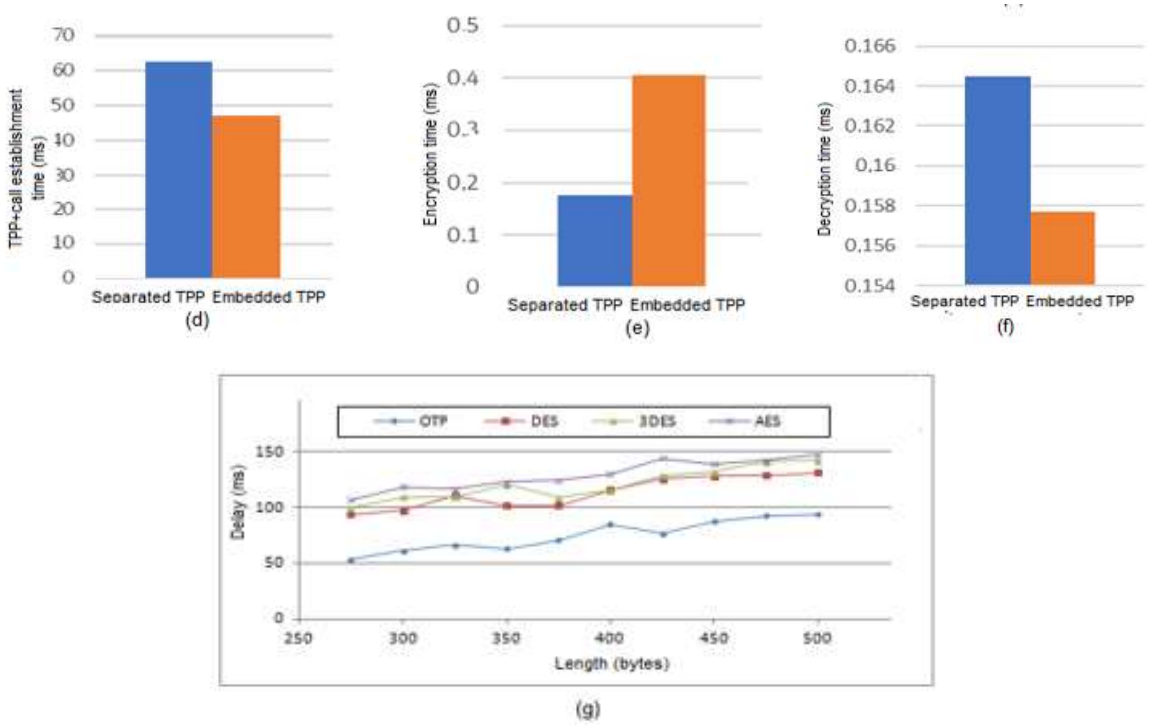

Figure 3. Round-trip-time and the increment reasons (continue)

Table 1. Compromised key impact

Method	Situation		
	Cracked TPP	First Packets	Other packets
Existing	Key is compromised	Compromised	Compromised
Proposed	Key is compromised	Compromised	No.

5. CONCLUSIONS

This work has proposed the insertion of TPP messages to transport the encryption shared key into TCP header or simply referred to as TCP-embedded three pass protocol. The purpose of TPP message insertion is to reduce key exchange time and to enable dynamic encryption key, where each packet in different window uses a different encryption key. The assessment was conducted by using socket programming and by modeling TCP connection uses UDP datagram. The evaluated network is assumed lossless. The evaluation results demonstrate that TCP-embedded three pass protocol outperforms the existing TPP in term of key exchange delay and the dynamic key capability. The proposed method is excellent in reducing shared key exchange delay up to 24.8% in average. Although round-trip-time increases, the increment is much lower than the key exchange process. In existing work, as for 10000 packets, once a key is leaked, whole packet is affected. The proposed method is able to protect data from the compromised key as dynamic key is exchanged frequently. If one key is compromised, the other packets are not accessible as they have their own key. Even dough channel is naturally lossy, this paper focused to assess the capability of TCP header carrying TPP messages. In other to evaluate the performances on lossy channel, ACK timer and memory should be added to manage packet retransmission. Additional solution is needed for possible duplicate TPP messages that are part of future work.

ACKNOWLEDGEMENTS

This research has been funded by Universitas Sumatera Utara through the Basic Research Schema of TALENTA 2020.

REFERENCES

[1] Goodman-Deane, J., *et al.*, "The impact of communication technologies on life and relationship satisfaction," *Computers in Human Behavior*, vol. 57, pp. 219–229, 2016, doi: 10.1016/j.chb.2015.11.053.

[2] G. Minopoulos, *et al*., "Comparison of video codecs performance for real-time transmission," *2nd IEEE Int. Conf. on Computer Communication and the Internet*, 2020, pp. 110-114, doi: 10.1109/ICCCI49374.2020.9145973.
[3] A. Langley, *et al*., "The quic transport protocol: Design and internet-scale deployment," *In Proc. of the ACM Special Interest Group on Data Communication Conf*, 2017, pp. 183–196, doi: 10.1145/3098822.3098842.
[4] F. Q. A. Al-Yousuf, and D. Roshidi. "Review on secured data capabilities of cryptography, steganography, and watermarking domain," *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, vol. 17, no. 2, pp. 1053-1059, 2020, doi: 10.11591/ijeecs.v17.i2.pp1053-1058.
[5] R. Patel, *et al*., "P-TORA: A TORA Modification Under TCP E2E-NewReno Model." *Int. Conf. on Futuristic Trends in Network and Communication Technologies*. Springer, Singapore, 2018, doi: 10.1007/978-981-13-3804-5_25.
[6] D. Djenouri, K. Lyes, and N. B. Algiers. "A survey of security issues in mobile ad hoc and sensor networks*." IEEE Communications surveys & tutorials*, vol. 7, no. 4, pp. 2-28, 2020, doi: 10.1109/COMST.2005.1593277.
[7] R. Illera, S. Ortega, and M. Petychakis, "Security and privacy considerations for cloud-based services and cloudlets," 2013. [Online], Available: https://cordis.europa.eu/docs/projects/cnect/3/317883/080/deliverables/001-OPENiD23SecurityandPrivacyConsiderationsforCloudbasedServicesandCloudlets.pdf.
[8] H. Takabi, T. Z. Saman, and B. J. James, "Mobile cloud computing and its security and privacy challenges," *In Security, Privacy, Trust, and Resource Manag. in Mobile and Wireless Com.*, pp. 384-407. IGI Global, 2014, doi:10.4018/978-1-4666-4691-9.ch016.
[9] H. Suo, *et al.*, "Security and privacy in mobile cloud computing," In *2013 9th Int. WCMC*, pp. 655-659, 2013, doi: 10.1109/IWCMC.2013.6583635.
[10] M. Mukherjee, *et al*., "Security and privacy in fog computing: Challenges," *IEEE Access*, vol. 5, pp. 19293-19304, 2017. doi: 10.1109/ACCESS.2017.2749422.
[11] L. Ximeng, Y. Yang, C. K. Raymond, and W. Huaqun, "Security and privacy challenges for internet-of-things and fog computing," *Wireless Com. and Mobile Comp.*, vol. 2018, 2018, doi: 10.1155/2018/9373961.
[12] A. M. Kakhki, *et al*., "Taking a long look at QUIC: an approach for rigorous evaluation of rapidly evolving transport protocols," In *Proc. of the 2017 Int. Measurement Conference*, 2017, pp. 290-303, doi: 10.1145/3131365.3131368.
[13] S. Nithya, K. Meena, "Genetic algorithm based bacterial foraging optimization with three-pass protocol concept for heterogeneous network security enhancement," *J. of comp. science*, vol. 21, pp.275-282, 2017, doi: 10.1016/j.jocs.2017.03.023.
[14] H. Boujezza, K. Hella, and A. S. Leïla, "Protection of IoT transaction using ID-KEM based on three-pass protocol countermeasure," In *2017 13th Int. Wireless Comm. and Mobile Computing Conf., IEEE*, 2017, pp. 423-428, doi: 10.1109/IWCMC.2017.7986323.
[15] P. Arulmozhi, J. Rayappan, and R. Pethuru, "A lightweight memory-based protocol authentication using radio frequency identification (RFID)," In *Adv. in Big Data and Cloud Comp*., pp. 163-172, Springer, Singapore, 2019, doi: 10.1007/978-981-13-1882-5_14
[16] D. Rachmawati, M. A. Budiman, "An implementation of the H-Rabin algorithm in the shamir three-pass protocol," *2nd Int. Conf. on Automation, Cognitive Science, Optics, Micro Electro-Mechanical System, and Information Technology*, 2017, doi: 10.1109/ICACOMIT.2017.8253381.
[17] S. Utama, "Three-Pass Protocol Concept in Hill Cipher Encryption," *Seminar Nasional Aplikasi Teknologi Informasi (SNATi)*, pp. 31-35, 2016.
[18] A. Subandi, R. Meiyanti, S. Mestika, R. W. Sembiring, "Three-pass protocol implementation in vigenere cipher classic cryptog-raphy algorithm with keystream generator modification," *ASTES Jour.*, vol. 2, no.5, pp. 1-5, 2017, doi: 10.25046/aj020501.
[19] P. Agung, S. Efendi, "Improving one-time pad algorithm on Shamir's three-pass protocol scheme by using RSA and ElGamal algorithms," *IOP Conf. Series: Journal of Physics: Conf. Series*,vol. 1235, p. 012007, 2019, doi: 10.1088/1742-6596/1235/1/012007.
[20] A. Badawi, M. Zarlis, "Impact three pass protocol modifications to key transmission performance," *IOP Conf. Series: Journal of Physics: Conf. Series*,vol. 1235, 2019, doi: 10.1088/1742-6596/1235/1/012050.
[21] A. Langley, *et al*., "The quic transport protocol: Design and internet-scale deployment," *In Proc. of the Conf. of the ACM Special Interest Group on Data Communication*, 2017, pp. 183-196, doi: 10.1145/3098822.3098842.
[22] U. Banerjee, *et al*., "eeDTLS: Energy-efficient datagram transport layer security for the Internet of Things," In *GLOBECOM 2017*, pp. 1-6, 2017, doi: 10.1109/GLOCOM.2017.8255053.
[23] E. Rescorla, and D. Tim, "The transport layer security (TLS) protocol version 1.3," 2018. [Online]. Available: https://tools.ietf.org/html/rfc8446
[24] M. Al-Akaidi, "Increasing uplink broadband video streaming protocol performance in WiMAX network," *Int. Jour. of Internet Protocol Technology*, vol. 7, no. 3, pp. 176-185, 2013, doi: 10.1504/IJIPT.2013.055448.
[25] G. Fairhurst, B. Trammell, and M. Kühlewind, "Services provided by IETF transport protocols and congestion control mechanisms," *RFC Series*, vol. 8095, 2017, doi: 10.17487/RFC8095.
[26] W. Shbair, T. Cholez, J. Francois, I. Chrisment, "A multi-level framework to identify HTTPS services," In *2016 IEEE/IFIP NOMS*, pp. 240-248, 2016, doi: 10.1109/NOMS.2016.7502818.
[27] M. Al–Akaidi *et al*., "An efficient negative acknowledgement–based transport protocol in 802.11 media streaming," *IJAHUC*, vol. 16, no. 3, pp. 161-171, 2014, doi: 10.1504/IJAHUC.2014.064418.

[28] H. Abdurrahman, O. S. Sitompul, and M. Naemah, "UDP-Lite Enhancement Through Checksum Protection," *In IOP Conference Series: Materials Science and Engineering*, *IOP Publishing*, vol. 180, 2017, doi: 10.1088/1757-899X/180/1/012146
[29] A. Aleksandr, and N, M. Andreevich, "Methods and algorithms for pseudo-probabilistic encryption with shared key," *Trudy SPIIRAN*, no. 61, pp.119-146, 2018, doi: 10.15622/sp.61.5.

BIOGRAPHIES OF AUTHORS

Suherman is a Senior Lecturer in Electrical Engineering Department, Universitas Sumatera Utara. He completed B.Eng from this university in 2000 and obtained MSc and PhD from RMIT, Australia and De Montfort University, UK subsequently in 2009 and 2013. He also followed some courses in TU-Ilmenau, Germany and European Bioinformatics Institute, Cambridge, UK. His research interest covers communication network and protocols, computer network as well as applied communication techniques.

Deddy Dikmawanto is a network engineer at the information system center of Universitas Sumatera Utara, Indonesia. He completed his B.Eng from Institute Technology of Sepuluh November, Surabaya and M.Eng from Universitas Sumatera Utara. His research interest is in computer network and security.

Syafruddin Hasan is an associate professor within Electrical Engineering Department, Universitas Sumatera Utara. He finished his B.Eng, M.Eng and Ph.D from Universitas Sumatera Utara, Bandung Institute of Technology and University of Malaysia Perlis subsequently. His research interest is in power electronics and power quality.

Professor Marwan Al-Akaidi has published over 200 papers in national and international journals and conferences. He chaired the Research Wireless & Mobile communication research group from 2002-2011 at De Montfort University-UK, then built a mobile & wireless research group at AOU. Currently He is the Vice President for research at the American University in the Emirates. He is the IEEE UK & Ireland Chair for the SPC. Chairman of the Modelling & simulation Middle East Conference, co-founder of Eurosis. Editor-in-Chief & Associate Editor-In-Chief for more than 5 international journals.