Selective-Packet Encryption Scheme with Authentication for Real-time Video Streaming Session using AES-256-GCM

Jewn Batinga*, Luis Miguel Valladolid and Marnel Peradilla
Computer Technology Department, College of Computer Studies
De La Salle University, Manila, Philippines
**Corresponding Author: jewn_batinga@dlsu.edu.ph*

Abstract: The popularity of the video-streaming services raises demand in terms of security and less-latency. These multimedia data being transferred over an unsecured channel include sensitive and private information. This paper proposes a secure encryption scheme that selectively encrypts video chunk payload using an Elliptic-curve Diffie–Hellman-negotiated session key. The proposed scheme also provides security and authentication by using the AES-256-GCM encryption method to generate an encrypted packet with authentication. This paper evaluates the latency delays caused by the encryption algorithm, computing load consumption, and end-to-end transmission latency for a real-world scenario. As a result, the proposed scheme aims to decrease the encryption latencies while providing strong session security and authenticity.

Key Words: selective-packet encryption; authentication; real-time streaming; AES; ECDH

1. INTRODUCTION

With the rapid rise of Internet services such as video streaming, Internet traffic has evolved from being pure text to multimedia. Multimedia sessions such as Voice-over-Internet Protocol (VoIP), video conferencing, and video surveillance feed requires enhanced security to prevent unauthorized users from accessing private and sensitive information. Providing end-to-end security for users is one of the challenging issues in network communications to maintain session confidentiality, privacy, and integrity.

The network technology industry standard for the protection of online information storage and transmission has evolved in the form of Virtual Private Networks (VPNs). The Internet Protocol Security (IPSec), a point-to-point secure tunnel, is used to create virtual networks and can be configured to secure all the data transmission between the two end-users through full data encryption through a single session key (Frankel & Krishnan, 2011; Gleeson et al., 2000). While these techniques may solve sniffing-related security issues, they may not apply to low-powered mobile devices (Bhattacharjya e t al, 2020; Narayan et al, 2016). The overhead caused by the encryption algorithm used in these solutions may be reduced only if selected packets are encrypted, as opposed to full encryption.

The most common way to secure multimedia sessions is to encrypt real-time protocol (RTP) voice/video packets using a single session key which is generated from symmetric encryption exchange (Velan et al., 2015). RTP defines the standardized packet format for the packets used for real-time multimedia services such as VoIP and video teleconference applications (Schulzrinne et al, 2003). RTP prioritizes delivery speed over data integrity, making it suitable for streaming services that require speed. RTP uses User Datagram Protocol (UDP) for the transmission of packets since UDP is faster than Transmission Control Protocol (TCP).

Existing proposed solutions use symmetric encryption key-approach using Data Encryption Standard (DES) and Advanced Encryption Standard (AES) (Gorbenko et al., 2018; Memos and Psannis, 2016). The session packet bitstream is encrypted with a session key, which is shared between the two parties before the actual data exchange.

The National Institute of Standards and Technology (NIST) introduced the AES-GCM (Galois /Counter Mode), an authenticated encryption algorithm that provides both confidentiality and authentication (Gueron et al., 2017; Rezvani and Diehl, 2019). This method is efficient since the data exchange does not two separate algorithms for authenticity and encryption.

On other hand, selective packet encryption helps decrease the latency of the multimedia transmission since fewer packets are to be encrypted. The method adds security to the multimedia transmission compared to other schemes since each of the selected packets will be encrypted with a unique secure-key that will only be used once. Several solutions have been proposed based on selective encryption. Jung et al. (2013) used selective packet encryption on RTP packets for real-time multimedia VoIP packets. Selective packets are encrypted differently with a one-time packet key, making it hard for the attackers to access other packets once they get to decrypt a packet. Diffie-Hellman key (DH-key) exchange procedures repeat for the selected RTP packets. Almasalha et al. (2008) proposed a similar scheme on low-powered mobiles. This study results in robust security with greater than 128-bit encryption per packet. However, this scheme is not recommended for higher resolution videos because of computing load issues.

The use of the single session key does not secure protection against brute-force attacks where the attacker captures a particular packet during transmission. In this paper, we intend to apply Elliptic-curve Diffie–Hellman (ECDH) key agreement procedure to generate a session key between communicating parties. The calculated shared key will be used as 1) encryption key, 2) seed to generate a unique nonce value, Initialization Vector (IV), and 3) seed to generate an authentication tag for packet integrity. The proposed selective-packet encryption scheme with authentication allows the session to encrypt a video packet with the shared key and additional integrity verification method at the receiver side. This paper aims to secure a live streaming session while maintaining the encryption latency values and computing load requirements to a minimum.

2. METHODOLOGY

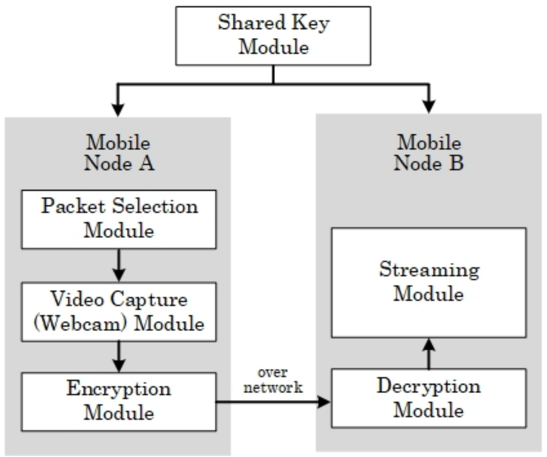

Fig. 1. Proposed authenticated selective-packet encryption scheme design overview

This section explains the methods in achieving the proposed selective-packet encryption scheme with packet authentication for video packets. Figure 1 shows the system overview of the proposed scheme with two communicating mobile nodes, MN_A and MN_B. The mobile nodes need to establish a session for video streaming. MN_A uses video capture hardware to generate the video frames to be sent to MN_B. The MN_B's streaming module allows to play and view the encrypted video stream by MN_A.

2.1 Session key agreement procedure

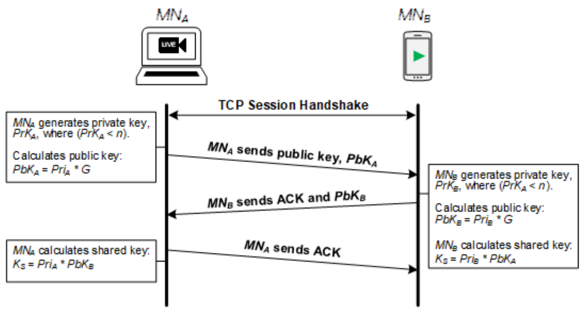
MN_A
LIVE
MN_B
TCP Session Handshake
MN_A generates private key, PrK_A, where ($PrK_A < n$).
Calculates public key: $PbK_A = Pri_A * G$
MN_A sends public key, PbK_A
MN_B generates private key, PrK_B, where ($PrK_A < n$).
Calculates public key: $PbK_B = Pri_B * G$
MN_B sends ACK and PbK_B
MN_B calculates shared key: $K_S = Pri_B * PbK_A$
MN_A calculates shared key: $K_S = Pri_A * PbK_B$
MN_A sends ACK

Fig. 2. Session key agreement process

Before the actual streaming of the encrypted video packets from MN_A to MN_B, the mobile nodes need to perform session key agreement for the session key, K_s. As shown in Fig. 2, MN_B establishes a TCP connection with MN_A's TCP socket. The proposed scheme uses the ECDH key agreement procedure to negotiate the session key, K_s between MN_A and MN_B. The ECDH defines G as an elliptic curve generator point. MN_A selects a random number, PrK_A to calculate its public key, $PbK_A = PrK_A \times G$. Then, MNA shares the calculated PrK_A to MN_B. Similarly, MN_B calculates its public key, PrK_b using its generated private key, PbK_B. After obtaining the public keys for the two communicating parties, MN_A and MN_B calculates the session key,

$$K_s = PrK_A \times PbK_B = PrK_B \times PbK_A \qquad \text{(Eq. 1)}$$

where PbK_A and PbK_B are the calculated public keys from the generated private keys PrK_A and PrK_B, respectively.

2.2 Selective-encryption procedure

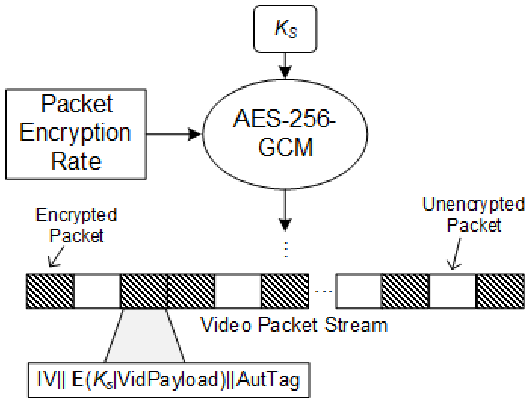

Fig. 3. Selective-encryption process

Recall, our proposed scheme only encrypts selected video packets for a given session. As shown in Fig. 1, the Packet Selection module decides the encryption rate for the packets. For example, if the encryption rate for 1000 video packets is 50%, the total encrypted packets are 500. The packet selection module determines whether the current payload must be encrypted or not. We added a single bit field to identify the encrypted packet. The encryption rate will be compared to the generated random number from the packet selection module. If the generated random number is less than or equal to the encryption rate, the encryption process will proceed, and the bit field will be set to 1. On the other hand, if the generated random number is greater than the encryption rate, the bit field will be set to 0.

The Elliptic Curve Key Generator Function (EC-KGF) is used in the ECDH. The primary function of EC-KGF is to generate keys that are intended to be used as a nonce value, IV_i for a particular video packet payload, $VidPayload_i$. Fig. 3 shows the AES-256-GCM process to generate the random block cipher, and the authentication tag (AutTag) using the nonce value IV and session key, K_s. The AES-256-GCM provides both message confidentiality and authenticity. The AutTag is calculated using the GHASH function of the AES-256-GCM encryption algorithm.

The encrypted video packet consists three fields: IV, encrypted video payload using session key and authentication tag.

2.3 Decryption procedure

When the video packets arrive at the MN_B, the MN_B's decryption module filters the packets to identify the encrypted packets. If the packet is not encrypted, the packet proceeds to the streaming module which plays the video packets (see Fig. 1).

If the encryption bit field is equal to 1, the decryption module proceeds to extract the original payload. The nonce value, IV_i, and the shared session key K_s will be used to decipher the unique packet key. Then, MN_B computes the $AutTag_i$ using the IV_i and K_S using the AES-GCM method. To check the integrity of the payload, the extracted authorization tag, AutTag from the encrypted packet will be compared to the MN_B's computed $AutTag_i$. After this process, the extracted payload proceeds streaming module for playing.

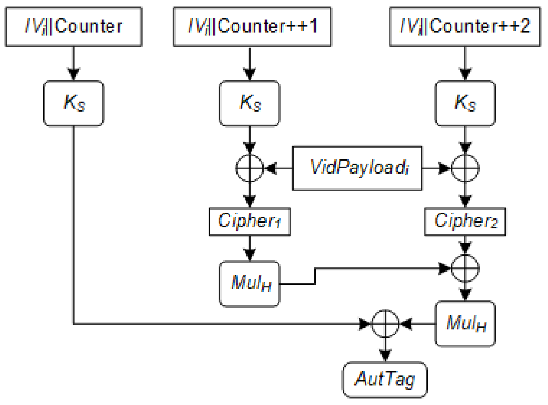

Fig. 4. AES-256-GCM encryption algorithm for confidentiality and authentication

3. RESULTS AND DISCUSSION

In this section, we evaluate the proposed authenticated selective-packet encryption scheme. We developed a web application for both streamer and receiver, to captures the camera feed of 720p resolution and play the video stream, respectively. The two mobile clients both run with the following computing specifications: 1.60GHz Intel Core i5-8250U and 8GB 1867 MHz DDR3 RAM. The two mobile nodes are connected via IEEE 802.11ac WLAN access point with a maximum transmission rate of 750 Mbps at 2.4 GHz. To evaluate the latencies, we limit a single session consists of 500 MJPEG-encoded RTP packets with 95000 bytes for each length. Also, we varied the encryption rate for different key sizes (8, 16, 32 characters) to the following: 25%, 50%, and 75%. All results shown in the succeeding tables and figures are already an average value of ten (10) trials for each test category.

3.1 Encryption and decryption latency

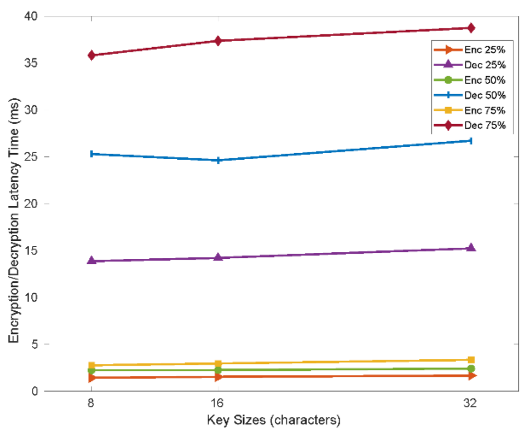

Fig. 5. AES-256-GCM encryption algorithm for confidentiality and authentication

The delay introduced by the AES-256-GCM algorithm for both ends need to be investigated especially for the real-time multimedia sessions. The encryption is done after the packet generation by the webcam capture module (Refer to Fig. 1). Therefore, the total latency to secure the entire session (L_{Enc}) is the sum of the latency to generate the video packet (L_{Pktgen}) and the latency to encrypt the video chunk (L_{PktEnc}), that is, $L_{Enc} = L_{Pktgen} + L_{PktEnc}$.

For the receiver point-of-view, the packet decryption is done after the computation and extraction of the corresponding packet key by the packet decryption module (Refer to Fig. 1). Here we defined the total decryption latency as L_{Dec}.

Using the developed web application, we run 10 live stream sessions for each encryption rate (25%, 50%, 75%) to measure the delay caused by the AES-256-GCM algorithm. In this experiment, we use different three different key sizes with 8, 16, and 32 characters. Take note that each character is equivalent to 4 bits. As shown in Fig. 5, the algorithm latency for both encryption and decryption increases as the session key size increases. For the minimum key size of 8 characters where 25% of the total packets are encrypted, the encryption and decryption latencies yield values of 1.44 milliseconds and 13.87 milliseconds, respectively. While for the maximum key size of 32 characters and 75% encryption rate, the encryption and decryption latencies are 3.35 milliseconds and 38.75 milliseconds, respectively. As a visual observation in the experiment, the key size of 32 and packet encryption rate of 75% introduced an intermittent play buffering on the receiver side. This means that the session key size should be limited to 16 and below with the packet encryption rate of 50% to satisfy the acceptable latency and video quality. Table 1 shows the summary of the encryption and decryption algorithm latency experiment for the different key sizes and encryption rates.

Table 1. Encryption and decryption algorithm latency caused by the proposed scheme

Key Size	25% Encryption Rate		50% Encryption Rate		75% Encryption Rate	
	L_{Enc} (ms)	L_{Dec} (ms)	L_{Enc} (ms)	L_{Dec} (ms)	L_{Enc} (ms)	L_{Dec} (ms)
8	1.44	13.87	2.25	25.30	2.77	35.82
16	1.54	14.22	2.26	24.63	2.96	37.37
32	1.67	15.24	2.42	26.72	3.35	38.75

3.2 Encryption algorithm memory consumption

Table 2. Memory (RAM) usage in megabytes (MB)

Key Size	25% Encryption Rate	50% Encryption Rate	75% Encryption Rate
8	19.39 MB	22.52 MB	25.88 MB
16	22.85 MB	24.39 MB	25.29 MB
32	20.34 MB	24.49 MB	24.16 MB

To have an overview of the CPU computation load of the proposed scheme, we measure the CPU usage of the sender side while the session is using a key size of 32 at 50% encryption rate. The CPU usage of the program reached 5.32% and 14.53% for the lowest and highest peaks, respectively. Take note that the computing machine is not running other heavy applications in the background.

Moreover, Table 2 summarizes the average memory usage in megabytes for the different test cases. The results show that the proposed scheme consumes a minimal amount of memory during a session. This means that our proposed scheme can run even for low-powered mobile devices.

3.3 Public network simulation

In this test evaluation, we implemented a testbed to simulate the real-world scenario that causes latencies introduced by the intermediate network devices such as routers. We set up a virtual machine (VM) to simulate a throttle bandwidth (BW) connection speed of 5 Mbps at the receiver side. In this test, both clients operate in a single machine. The receiver side runs in the created VM which is connected to the virtualized network. The sender side (streamer) runs on the installed OS in the machine. The VM computing specifications are the following: 2-Core processor with a 1 GB RAM. Similarly, we test different key sizes and encryption rates to evaluate the end-to-end latency during a session. The total end-to-end latency (L_{E2E}) is the sum of the transmission time (L_{Trans}), the latency to encrypt the video packet (L_{Enc}), and the latency to decrypt the video packet (L_{Dec}), that is, $L_{E2E} = L_{Trans} + L_{Enc} + L_{Dec}$. We compare the latency delay caused by the simulated public network to a point-to-point connection via a local wireless network.

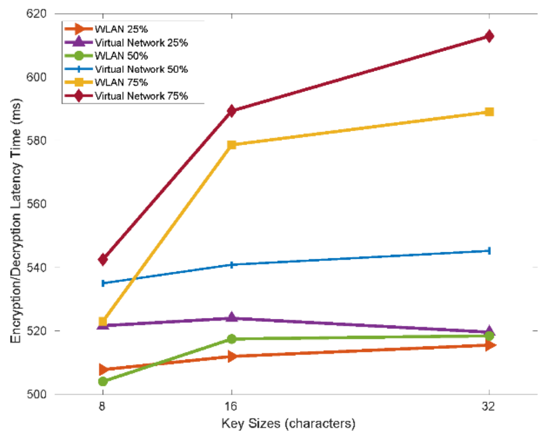

Fig. 6. End-to-end latency comparison on simulated public network (BW = 5 Mbps) vs. WLAN

Fig. 6 shows that the point-to-point connection of WLAN is obviously faster than the virtualized network for all the test variations. For the key size of 32 at 75% encryption rate, the test yields 612.88 milliseconds of end-to-end latency on simulated public network. The average L_{E2E} difference between the network setup using key sizes of 8 and 16 are 21.42 and 15.36 milliseconds, respectively. For the key size of 32, the average difference of L_{E2E} between the two network modes is 18.24 milliseconds. At 75% encryption rate with a 32-character key size, the L_{E2E} is 612.88 milliseconds. This delay is still acceptable in transmitting low-resolution video frames. However, high-definition video feed operating with a higher encryption rate may introduce intermittent playing buffering and pixelized video frames. We summarized the end-to-end latency in simulated public network and WLAN in Table 3.

Table 3. End-to-end latency in WLAN and virtual network environment

Key Size	25% Encryption Rate		50% Encryption Rate		75% Encryption Rate	
	WLAN	Virtual Network	WLAN	Virtual Network	WLAN	Virtual Network
8	507.85	521.69	504.11	535.06	523.07	542.53
16	512.02	524.03	517.52	540.90	578.66	589.34
32	515.58	519.66	518.50	545.27	589.01	612.88

5. CONCLUSIONS

This paper proposed a selective-packet encryption scheme for real-time video streaming sessions, which is designed to achieve confidentiality and maintain the integrity of the encrypted video packets. Using the ECDH key exchange procedure, a session key is negotiated to serve as a seed to produce the unique random nonce value and an authentication tag and to encrypt the video chunk payload. This proposed scheme contributes to minimizing the encryption latencies for real-time multimedia services by implementing selective-packet encryption.

Our analysis showed that the proposed scheme introduces a small number of latency values for all key sizes as long as the encryption rate is below 75%. The scheme also consumes minimal RAM allocation for all test cases, which can be adapted by low-powered mobile devices. To simulate the real-world latency and delay, we created a testbed using a virtual machine to evaluate the end-to-end transmission delay introduced by the scheme. The results showed that even using the 32-character key size and 75% encryption rate, the end-to-end transmission for the simulated public network is still in acceptable values for transmitting low-resolution video frames. This proved that the proposed scheme does not affect the real-time performance of the video stream while providing strong security for the session

6. REFERENCES

Almasalha, F., Agarwal, N., & Khokhar, A. (2008, December). Secure multimedia transmission over RTP. In 2008 Tenth IEEE International Symposium on Multimedia (pp. 404-411). IEEE.

Bhattacharjya, A., Zhong, X., Wang, J., & Li, X. (2020). CoAP—application layer connection-less lightweight protocol for the Internet of Things (IoT) and CoAP-IPSEC Security with DTLS Supporting CoAP. In Digital Twin Technologies and Smart Cities (pp. 151-175). Springer, Cham.

Frankel, S., & Krishnan, S. (2011). IP Security (IPsec) and Internet Key Exchange (IKE) Document Roadmap. RFC 6071.

Gleeson, B., Lin, A., Heinanen, J., Armitage, G., & Malis, A. (2000). A Framework for IP Based Virtual Private Networks. RFC 2764.

Gorbenko, I., Kuznetsov, A., Tymchenko, V., Gorbenko, Y., & Kachko, O. (2018, October). Experimental studies of the modern symmetric stream ciphers. In 2018 International Scientific-Practical Conference Problems of Infocommunications. Science and Technology (PIC S&T) (pp. 125-128). IEEE.

Gueron, S., Langley, A., & Lindell, Y. (2017). AES-GCM-SIV: Specification and Analysis. IACR Cryptol. ePrint Arch., 2017, 168.

Jung, Y., Festijo, E., & Atwood, J. W. (2013). Securing RTP Packets Using Per-Packet Key Exchange for Real-Time Multimedia. ETRI Journal, 35(4), 726-729.

Memos, V. A., & Psannis, K. E. (2016). Encryption algorithm for efficient transmission of HEVC media. Journal of Real-Time Image Processing, 12(2), 473-482.

Narayan, S., Ishrar, S., Kumar, A., Gupta, R., & Khan, Z. (2016, July). Performance analysis of 4to6 and 6to4 transition mechanisms over point to point and IPSec VPN protocols. In 2016 Thirteenth International Conference on Wireless and Optical Communications Networks (WOCN) (pp. 1-7). IEEE.

Rezvani, B., & Diehl, W. (2019). Hardware Implementations of NIST Lightweight Cryptographic Candidates: A First Look. IACR Cryptol. ePrint Arch., 2019, 824.

Schulzrinne, H., Casner, S., Frederick, R., & Jacobson, V. (2003). RTP: A Transport Protocol for Real-Time Applications. RFC 3550

Velan, P., Čermák, M., Čeleda, P., & Drašar, M. (2015). A survey of methods for encrypted traffic classification and analysis. International Journal of Network Management, 25(5), 355-374.