Nat. Volatiles & Essent. Oils, 2021; 8(4): 775-786

NVEO
Natural Volatiles & Essential Oils

Efficient Data Origin Authentication Scheme for Video Streaming Transmitted by Multiple Senders

Namhi Kang [1]

[1]Duksung Women's University, Seoul, Korea

*Corresponding author. Email: kang@duksung.ac.kr

Abstract

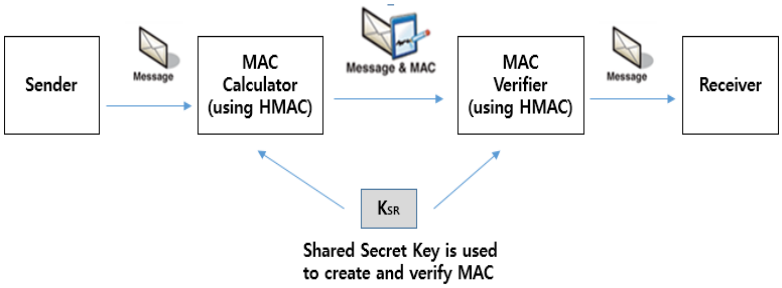
This study proposes a scheme to authenticate real-time streaming data and verify the integrity of the transmitted data in a service application with multiple senders or receivers. This data origin authentication scheme provides advantages in performance compared with TESLA, a multicasting data origin authentication scheme widely applied in conventional authentication techniques. That is, it can authenticate rapidly received data, even if resource-restricted devices, such as CCTVs and drones, are equipped with fewer computing resources and memory.

Keywords: Streaming Authentication, CCTV, Drone Security, Key Agreement Scheme

1. Introduction

On the Internet, we mainly compute and attach a message authentication code (MAC) to the data to be transmitted to authenticate the message and verify the integrity of the transmitted data. The MAC is usually calculated by using HMAC or a public key-based digital signature. Among the two methods, a public key-based digital signature requires a large amount of computation, resulting in a long delay for generating and verifying signatures, and also demands many hardware resources (high-performance CPU, memory space). Considering these issues in terms of performance, we mainly use HMAC, a hash-based message authentication code [1, 2]. Fig. 1 shows how to calculate a MAC using HMAC and add the calculated MAC to a message [3].

Figure 1. Data Origin Authentication Scheme Using HMAC



As shown in Fig. 1, when applying HMAC, the sender generates a MAC (the authentication code) using the message and secret key to be transmitted as input for the HMAC function. Upon receiving this message, the receiver verifies it by calculating the MAC through the HMAC function using the message received and the secret key shared with the sender in advance. That is, the message's integrity and authenticity are verified if the HMAC calculated by the receiver and the MAC sent by the sender are the same. This method takes advantage of the fact that an entity without a shared secret key cannot create or verify a MAC.

Although HMAC is superior in performance to the public key-based digital signature method, there are difficulties in applying HMAC in applications that require real-time characteristics, such as audio or video transmission, or when multiple senders and receivers exchange messages [4]. An example of an n:1 communication method with multiple senders is an application that transmits video data taken by CCTVs or drone clusters to a single source, such as a control system. On the other hand, an example of a 1:n communication method is when video streaming from a single source is broadcast to multiple users.

In terms of directly applying HMAC to the two communication methods above, a problem occurs because the multiple senders or receivers must share different secret keys [5]. For example, when a single sender transmits a message to 10 receivers, the sender must securely share and store a secret key with each of the 10 receivers in advance. The sender must also calculate the HMAC 10 times (by applying each secret key) to send a different MAC to each receiver. When one shared secret key is applied to all of the video streaming, security is reduced because only one of the devices in the group can be authenticated. Furthermore, a key management method that requires generating and distributing different keys for multiple receivers is not easy. In particular, key management becomes more difficult because the secret key needs to be changed after a certain period to ensure forward security. CCTVs and drones have limited resources (CPU, memory, etc.) compared with general Internet systems, so there are difficulties in applying secret key sharing techniques frequently. Therefore, key sharing and frequent MAC computations are not easy for resource-restricted devices [6, 7, 16].

To solve this problem, this study proposes a method to authenticate the data transmitted by resource-restricted devices in real-time and verify the integrity of the transmitted data.

This article is structured as follows. Chapter 2 describes conventional methods to compare with the scheme proposed in this study. In particular, this study considers TESLA [8], which is widely applied in conventional streaming authentication technologies. Chapter 3 describes the reasons why previously proposed methods are not suitable for service environments, such as CCTVs and drones. Chapter 4 describes the proposed authentication scheme, and the final chapter analyzes its stability and performance.

2. Related Work

The secret key sharing scheme proposed in this paper, on one hand, can be used to provide integrity and confidentiality of real-time video streaming transmitted by multiple resource constrained devices. Streaming data is widely used and it is an important form of data in various fields, especially in the field of industrial control and automation [9]. In such a scenario, most devices, which are generally used as data streaming sources, are resource constrained devices, thus the devices send their streaming data to a powerful cloud for keeping and processing required high computing and storage resources [10, 11, 12]. Therefore, an efficient key sharing scheme is highly required for resourse constraiend devices to use the shared secrete key for their streaming data authentication method.

On the other hand, the data origin authentication method which utilize the key sharing scheme proposed in this paper can be applied for CCTV networks or drone networks, where multiple data sources and receivers are used. In these cases, the sending and receiving nodes are lightweight devices that are difficult to perform complex operations and mainly use wireless networks. Therefore, it is better to apply HMAC than the digital signature method, which is mainly applied to wire-based Internet systems [4, 5].

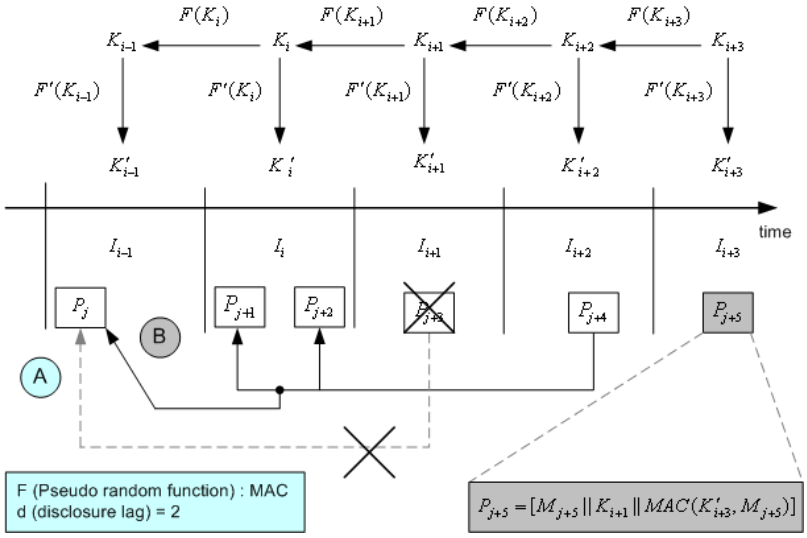
This chapter describes several authentication techniques proposed for wireless networks, starting with Timed Efficient Stream Loss-tolerant Authentication (TESLA), the basic multicast data origin authentication

applied to each of these techniques [3, 8]. In particular, the description focuses on the secret key sharing scheme applied in TESLA.

TESLA allows all receivers to check the integrity and authenticate the source of data streams, and is designed for video streaming broadcasting in a 1:n multicasting environment. TESLA uses hash chain technology to reduce frequent session key sharing and solve 1:n different key sharing. That is, a message authentication code (MAC) is calculated and attached to a video data packet that generates and transmits a secret key using a reverse hash chain. Therefore, it is possible to authenticate the message sent to all receivers with a single MAC, even if there are multiple receivers.

To apply TESLA safely, the sender and multiple receivers need to set the initial secret key applied to the one-way hash chain technology in advance. As shown in the fig. 2 below, the sender obtains a secret key that is valid for a specified time interval (interval I) from the hash chain, calculates the MAC to which this secret key is applied, and uses it as the authentication value for the packet to send.

Figure 2. Data origin Authentication in TESLA



As shown in Fig. 2, the secret key used in TESLA is created in advance through a one-way keychain (i.e., $F(K)$). The key used for each interval is disclosed after a certain interval according to a predetermined schedule (i.e., d: disclosure lag). In TESLA, the sender transmits the secret key that calculated the MAC at time t to the receivers after a certain period of time. Therefore, each receiver verifies the MAC after waiting for the private key to be disclosed to check the integrity and authenticate the received message. As shown in the example in Fig. 2, the secret key used in interval I_i is sent to the receivers in I_i+2 after 2 intervals. That is, the receivers can verify the authenticity of the video data received after 2 intervals. Therefore, delays are inevitable for verification.

During the data origin authentication process, the MAC for packets P_{i+1} and P_{i+2} sent in interval I_i is created using K_i, and K_i is disclosed to the receivers through P_{i+4}, which is sent after two time intervals determined by the value of d. Therefore, the receiver verifies the authenticity of the packet received after a predetermined time interval. Due to the one-way computational characteristic of the secure hash function, an attacker cannot exploit the key used before the sender discloses the key value that created the MAC.

That is, only the K_{i-1} value is disclosed in interval I_i, and the MAC value of P_{i+1} can only be changed by calculating K_i with the value of K_{i-1}.

Message authentication schemes such as LHAP [13], LAP [14], HEAP [15] have been proposed to authenticate data in mobile ad hoc networks (MANET). These schemes provide a way to verify data transmission on a hop-by-hop basis between nodes without distinguishing between control and data messages in a MANET.

LHAP and LAP are based on one-way chain technologies applied to the most widely used TESLA method, among multicast data transmission authentication techniques. LHAP is a hop-based data authentication scheme that considers problems that may occur when applying TESLA to a MANET. For data transmission, two key chains (i.e., TRAFFIC key and TESLA key chain) are created in advance, and they are shared through public key-based authentication with neighboring nodes in hop #1 before communication. The sender transmits a message in the format of Eq. (1) below.

$$\mathrm{A} \rightarrow * : \mathrm{M}, {\mathrm{K_A}}^{\mathrm{F}}(\mathrm{i}) \quad (1)$$

In Eq. (1), * refers to the transmission to all neighboring nodes, and ${\mathrm{K_A}}^{\mathrm{F}}(\mathrm{i})$ is a value that can be authenticated through the TRAFFIC keychain. Therefore, when node A joins the MANET, it sends a message, as shown in Eq. (2), to the neighboring nodes as a verification value to be used in the authentication process.

$$\mathrm{Cert_A},\ \mathrm{Sign_A}[\ \mathrm{A} \mid {\mathrm{K_A}}^{\mathrm{T}}(0) \mid {\mathrm{K_A}}^{\mathrm{F}}(0) \mid {\mathrm{T_A}}^{\mathrm{T}}(0) \mid {\mathrm{T_A}}^{\mathrm{F}}(0)] \quad (2)$$

In Eq. (2), ${\mathrm{K_A}}^{\mathrm{T}}(0)$ is the verification value of the TESLA keychain used to update the TRAFFIC key in the KEYUPDATE message, and ${\mathrm{T_A}}^{\mathrm{F}}(0)$ and ${\mathrm{T_A}}^{\mathrm{T}}(0)$ represent the start time of each key.

The problem with LHAP is the possibility of man-in-the-middle and wormhole attacks. In particular, wormhole attacks can be easily made through the collaboration between internal and external attackers because the message itself does not act as a factor in creating the authentication values.

LAP is also a hop-based data authentication scheme specialized for MANET based on a one-way key chain. Unlike LHAP, LAP uses the message as a factor to compute the MAC value and operates based on one keychain instead of two. When node A joins a MANET for message authentication, it sends a message in the format shown in Eq. (3) to neighboring nodes.

$$\mathrm{A} \rightarrow *:\ \mathrm{Cert_A}, [\mathrm{A} \mid {\mathrm{h_n}}^{\mathrm{A}} \mid \mathrm{H_A}],\ \mathrm{Sign_A}(\mathrm{A}, {\mathrm{h_n}}^{\mathrm{A}}, \mathrm{H_A}) \quad (3)$$

where ${\mathrm{h_n}}^{\mathrm{A}}$ and $\mathrm{H_A}$ represent the verification value of the one-way keychain and the secure hash function to be used. Node A then sends data in the following format when a message needs to be sent. $\mathrm{A} \rightarrow \mathrm{B}$: $\mathrm{M}, \mathrm{MAC}(\mathrm{M}, {\mathrm{h_i}}^{\mathrm{A}})$, where M is the message to be sent, and ${\mathrm{h_i}}^{\mathrm{A}}$ is the key value currently used by node A, which is not yet disclosed. Therefore, node B must have ${\mathrm{h_i}}^{\mathrm{A}}$ sent via a KEYUPDATE message from node A to verify the authentication value.

LAP, like TESLA, uses a key value disclosed after a certain period of time to verify authentication, so the total delay experienced by the endpoint acts as a disadvantage. LAP is also vulnerable to DoS attacks because the received data must be stored until KEYUPDATE messages are received.

Unlike LAHP and LAP, HEAP performs hop-by-hop authentication by using HMAC or NMAC internal computation. When a node joins a MANET, it must share the iKey, which is a group key shared by all of the nodes, and the oKey, which is a separate secret key shared with neighboring nodes directly connected by

one hop. The two shared keys are used to compute the HMAC, as shown in Eq. (4) (i.e., inner key and outer key).

$$\mathrm{HMAC}(\mathrm{M}, \mathrm{K}) = \mathrm{H}(\mathrm{oKey} \mid \mathrm{H}(\mathrm{iKey} \mid \mathrm{M})) \quad (4)$$

As shown in Eq. (4), computing the HMAC in HEAP is efficient because long messages are calculated using only the iKey, and afterward, the final authentication value is calculated using the oKey. However, the major drawback of this approach is that if a message needs to be broadcast (e.g., sending routing control messages), each neighboring node has to compute the authentication value separately. In general, a MANET should also consider the environment in which nodes move. In the case of frequent movement, unlike LAHP and LAP, HEAP requires the use of a public key-based key sharing method, which requires a significant amount of computing to share keys with all neighboring nodes, which leads to poor performance.

3. Problem Statement

The scheme proposed in this study can be applied when targeting multiple receivers (i.e., 1:n), the traditional multicasting environment, and when multiple senders send video data to a single receiver (i.e., n:1). The applications include:

- When multiple drones are deployed in service application sites (e.g., applied by the police, broadcast shooting, etc.) to send video data to the GCS (Ground Control System) or control center in real-time
- When multiple receivers receive video data from a single drone
- When a control center receives video data from multiple CCTV cameras

The following problems occur when applying multicast authentication protocols such as TESLA to the environments above (drone or CCTV applications).

- Hash chains for generating secret keys need to be set up for all of the multiple senders and receivers (1:1) in advance
- The sender and receiver need time synchronization to compute the time required to disclose the secret key (necessary to calculate the key disclosure time)
- The secret keys on the hash chain need to be reset, if they run out too soon
- The transmitted data is authenticated after a certain period of time, so real-time transmission is difficult due to delays in transmission authentication by the receiver

The proposed scheme solves the problems above by enabling data origin authentication without generating a different secret key for each sender-receiver. Furthermore, the proposed technique has the following advantages over TESLA.

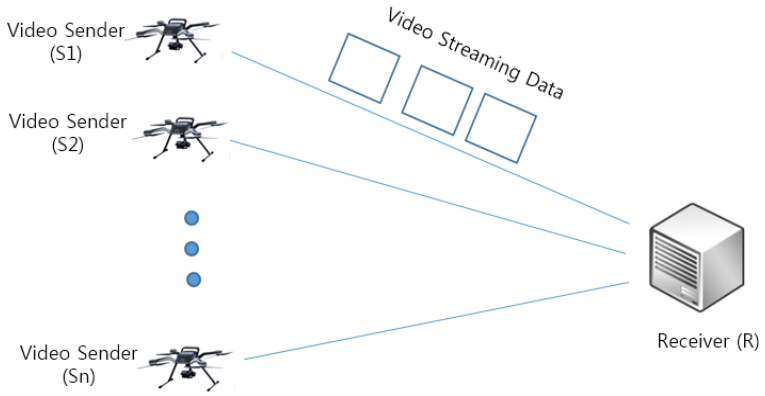
Does not depend on secret key hash chains, so the entities do not have to synchronize their time

- Enables immediate authentication compared with authentication methods that require time to disclose the secret key (less authentication delay)
- Hash chains can generate a fixed number of secret keys, but the proposed technique can create many types of secret keys, so there is minimal burden for resetting secret keys.

4. Proposed Scheme

The figure 3 below shows an example of an environment that applies the scheme proposed in this study (i.e., a service environment in which N senders transmit video data to one receiver).

Figure 3. Applied System Architecture



In the case of applying MAC to authenticate data in the example shown above, R must set and manage different private keys with each S. However, the proposed scheme can authenticate the data source and verify the integrity of transmitted data, even if R shares the secret key table with all senders.

4.1. Authentication and Secret Key Table Configuration/Reconfiguration

This study assumes that video data transmission and reception are based on safely sharing an initial master secret key (KSR) in advance. The size of the master secret key and the secret key for data origin authentication is assumed to be 128 bits, but longer secret keys can be created and applied to upgrade security.

The sender and receiver exchange the following messages to authenticate the transmitted data and verify the integrity of the transmitted data. The subject sending the message can be changed. That is, the sender who wants to initiate data transmission can send a request message to the receiver first. Or, the receiver may send a secret key configuration message to the sender to request video data transmission. This study is based on the latter. The proposed scheme can be used to configure a secret key table before initiating communication, and to reconfigure a secret key table during communication.

The figure below shows the flow of messages between the sender and receiver for authentication and secret key table configuration. The messages sent in each step and the abbreviations used in the figure are as follows.

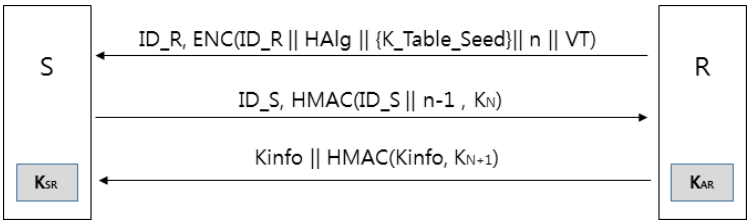
- ID_R, ID_S: Identifier for each receiver and sender
- HAlg: HMAC algorithm to be used for calculating MAC (e.g., HMAC using SAH_128)
- {K_Table_Seed}: Seed value, which is a set of numbers for configuring secret key table
- n: the number of key in the generated secret key table (a.k.a., table size)
- VT: Valid time for the generated secret key table
- Kinfo: Generation information of the secret key for calculating HMAC (see 4.2 for details)

• ENC(): Encryption algorithm (e.g., AES128)

• HMAC(): HMAC algorithm (e.g., HMAC using SAH_128)

A || B: Concatenation of message A and B

Figure 4. Function Flows



- The receiver (R) encrypts and sends its ID, the HMAC algorithm to be used, the secret key table seed, number n, and the valid time for the generated secret key table to the sender to request video data transmission.
- The sender (S) performs the following steps after a request for video data transmission from R.
- S decrypts the message sent in Step 1 to obtain ID_R, and compares this with the unencrypted ID_R to verify whether R shares the same KSR to authenticate the data source.
- A hash table based on {K_Table_Seed} is created. {K_Table_Seed} contains random numbers and time stamps at the time of the request. N secret keys based on these two pieces of information using a forward hash chain are created. That is, the first secret key is HASH (random number || time stamp), and the second secret key is the result of hashing the first secret key again. Therefore, the n-th secret key becomes hash (n-1th secret key). A secret key table consisting of n secret keys is created through this process. As a result, both the sender and receiver share the same n secret keys constituting the secret key table.
- The sender uses its ID and number n-1 as an input message, calculates the HMAC using the n-th secret key, and sends it to the receiver with its ID. The transmitted HMAC can be computed and verified only by the sender and receiver sharing the secret key table.
- R, the video data receiver, performs the following functions after receiving a message from S.
- The receiver completes the key table shared with the sender by verifying the received ID_S by using the n-th secret key of the secret key table calculated by itself. The receiver then authenticates the data source by verifying that the sender knows the same master key by confirming that the n-value sent in Step 1 was decrypted.
- The receiver uses Kinfo (refer to Section 4.2 for details), the session key generation information, to calculate the MAC of data to be transmitted in the future as an input message, and merges Kinfo with the MAC computed by using the n-1th secret key in the key table and sends it to the sender.
- After receiving the message from Step 3, the sender verifies the MAC with the n-1th secret key and the transmitted Kinfo. If there is no problem, the sender proceeds by deriving a session key to be used in the future through Kinfo, and completes creating the key table.

After completing the above process, the sender and receiver share a secret key table, as shown in the following example. For example, if the K_Table_Seed value is {1234567890 || 20180730121432 }, the first

part is a random number, and the latter part denotes the “year. month. day. hour. minute. second.” SHA1 was used as the hash algorithm, and this study uses the 128 bits from the front of the 160 bits generated by SHA1 as the secret key. Different hashing algorithms can be applied and the size of the secret key can be adjusted according to the settings. For example, K1 and K2 are calculated as follows, and Table 1 shows a secret key table configuration with 7 different sizes.

K1 = SHA1(123456789020180730121432)

K2 = SHA1(K1) = SHA1(2723d6bf30b98d4396ab327ee20d8399)

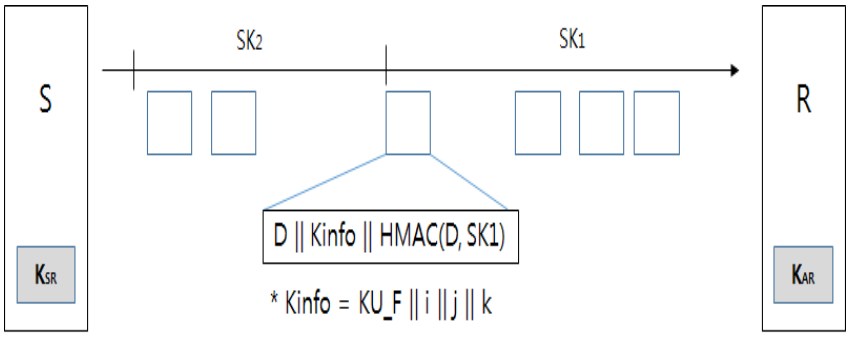
Table 1. Generated Secret Key Table

Secret Key Index	Secret Key (128 bits, hexadecimal notation)
K1	2723d6bf30b98d4396ab327ee20d8399
K2	f4b3089a5a59922ddc56f0b0306ac2dc
K3	04e45d7bcabbde40b38dff2a19232782
K4	0156d8768f76e48eccc864df765228d6
K5	43f13fc53bd58a9fe4368ed79a097a9e
K6	926a6d027b913774a3885e3153d3da7
K7	b79752970d070a1426175fc4d0a1612a

4.2. Secret Session Key Selection and Streaming Data Authentication

Video streaming transmission follows the process in the figure below to authenticate the data to be transmitted.

Figure 5. Proposed Authentication Scheme

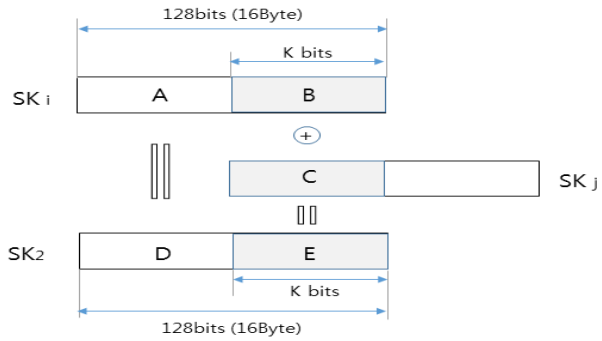


When video streaming after completing the secret key table steps described in Section 4.1, the MAC is computed and attached to the data using the HMAC function for data authentication.

The Kinfo transmitted with the data is used to derive session keys, and KU_F (Key Update Flag), the start bit of Kinfo, is used as a flag bit to choose whether to derive a new session key. If KU_F is 0, it means that the previously used session key will be used as is, and if KU_F is 1, it means that a new session key will be used after the current data. The example in the figure shows the case where S wants to change the session key from SK1 to SK2, while the S and R are computing the MAC to authenticate data transmission by deriving SK1 from the secret key table.

Kinfo consists of three numbers: i, j, and k. If KU_F is 0, all three numbers are zeros. If KU_F is 1, numbers i and j represent the secret keys in the secret key table. Both the sender and the receiver calculate Ski and SKj from the key table with i and j (i.e., the i-th hash value and the j-th hash value of K_Table_Seed in Section 4.1). The last number k refers to the latter part of the i-th key (marked as B in the figure) and the front k bits of the j-th key (marked as C in the figure).

Figure 6. Session Key Generation Method



As shown in the figure 6, the k bits at the back of SKi and the k bits at the front of SKj are calculated via the exclusive OR function to generate the k bits at the back of the new secret session key to be used (marked as E in the figure). The first part of the new session key (marked D in the figure) uses the value of A of the I-th secret key.

In summary, the final changed secret key SK2 is created by merging D and E. That is, SK2 = [D||E], where the size of E is k bits. The first part of SK2 (D) has a size of 128kbits and is replaced by the first part of the i-th key in the key table (marked A in the figure). The E part of SK2 has a size of k bits and uses the XOR operation values of B and C.

5. Performance and Security Analysis

This chapter analyzes the performance and security based on an example of applying the data origin authentication scheme for multiple sources proposed in Chapter 4.

5.1. Security Analysis

The sender using the proposed scheme encrypts and transmits the HMAC algorithm, secret key table seed, number n, and the valid time for the secret key table to be used in the future to provide data origin authentication and data integrity to the receiver. If both parties securely share the master key in advance, attackers cannot create the secret key table even if they eavesdrop on the messages. Various methods can be applied to set the master key securely according to the application environment and security policy.

In the proposed scheme, the sender and receiver can reset the session key in the secret key table whenever they want, so it is easy to provide forward security and respond to replay attacks or session key prediction attacks. The traditional LHAP system only adds a key for authenticating the data source to the transmitted authentication packet. Therefore, LHAP only guarantees the authentication of the transmitted key value but cannot authenticate the message and provide integrity. When transmitting data by the proposed method, the MAC is computed on the message and serial number to provide data origin authentication and integrity, and prevent replay attacks.

The following describes how to reset the secret key, assuming that the secret key table example in Chapter 4 is set and the Kinfo is [1, 3, 5, 16].

This example of the parties resetting the secret session key shows that the secret session key can be set securely. Based on the example in Section 4.2, the following shows how to reset the secret key. The party who wants to reset the session key requests the session key reset by setting the KU_F bit to 1 in Kinfo. Then, the two parties can compute a new session secret key through the i and j bits in the Kinfo field. For example, if i and j are 3 and 5, the secret session key is set as follows.

The sender and receiver retrieve the secret keys corresponding to K3 and K5 in the pre-set secret key table.

K3 = 04e45d7bcabbde40b38dff2a19232782

K5 = 43f13fc53bd58a9fe4368ed79a097a9e

The E part of the session key SK to be used between the two parties is the result of XORing the lower 16 bits of K3 (2782) and the upper 16 bits of K5 (43f1). The value obtained by converting 2782 and 43f1 to binary and XORing them is as follows.

(0010 0111 1000 0010) XOR (0100 0011 1111 0001) = 0110 0100 0111 0011 = 64e3

The D part of SK becomes 04e45d7bcabbde40b38dff2a1923 due to substituting the upper value of K3. Therefore, the final session key SK is as follows.

SK = 04e45d7bcabbde40b38dff2a192364e3

5.2. System Performance

The proposed scheme can solve the problems of conventional methods described in Chapter 3, and there are no difficulties in each sender and receiver having a different secret key. The proposed scheme can derive session keys equivalent to the size of the key length from a small number of hash chains through the method described in Section 5.1, thus using less memory than TESLA and the many techniques proposed based on it. In the case of TESLA, the key chain needs to be reset after creating a fixed number of secret keys with the hash chain. However, the proposed scheme provides performance benefits for application services using lightweight devices with limited resources because it can generate many types of secret keys and minimize the burden of resetting secret keys.

The proposed scheme is also better than convention methods in terms of authentication delay. It does not rely on a private key hash chain, so the parties do not have to synchronize their time. The scheme also enables immediate authentication compared with authentication methods that require the receiving party to wait before disclosing the secret key.

In summary, the performance advantages of the proposed scheme are as follows.

1. Reduced number of key hash chains stored by the sender and receiver

2. Fewer operations to set a new hash chain when the secret keys run out too soon

3. Secret keys that are as long as the key length can be derived from a small number of hash chains

4. No need for sender-receiver time synchronization

5. Enables immediate authentication due to using forward hash chains (no authentication delay)

6. Allows the sender and receiver to dynamically change the session key compared with the static method in which the secret key is used in a specified interval based on time synchronization (The level of safety is increased by excluding from the attacker information that secret keys are used for each interval. Moreover, fewer secret keys are consumed in the hash chain by using the same key if there is little data to be transmitted)

6. Conclusion

This study proposes a scheme to authenticate the data transmitted by resource-restricted devices in real-time and verify the integrity of the transmitted data. The proposed authentication scheme is suitable for application services that stream real-time from multiple senders to a single receiver, or applications or services where a single sender has to multicast real-time video streaming to multiple receivers.

The proposed scheme can authenticate multiple video data sources with limited available resources, such as drones and CCTVs. In this way, the receivers can only receive video data from legitimate senders. Each sender can also create different secret keys using a single secret key table. This not only improves security but also reduces the burden of the sender with limited available resources because there is no need to create and share new secret keys with the receiver at regular time intervals. Since the proposed scheme uses a secret key table to create secret keys, it can generate relatively more secret keys than methods that use hash chains to create secret keys, such as TESLA, even when using less memory.

Acknowledgments

This Research was supported by Duksung Women's University Research Grants 2019.

References

Junhui Li, Liji Wu, Xiangmin Zhang, “High-performance implementation of an HMAC processor based on SHA-3 Hash function,” Int. Conf., on EDSSC, 2017.

Lawrence, T., Li, F., Ali, I., Kpiebaareh, M. Y., Haruna, C. R., & Christopher, T. “An HMAC-based authentication scheme for network coding with support for error correction and rogue node identification,” ELSEVIER Journal of Systems Architecture, Vol. 116, June 2021

H. Krawczyk, M. Bellare, R. Canetti, “HMAC: Keyed-Hashing for Message Authentication,” IETF Std., RFC 2104, February 1997

N. Kang and C. Ruland, "DiffSig: Differentiated digital signature for real-time multicast packet flows", Proc. Trust Privacy Digital Business, pp. 251-260, 2004.

Diana Berbecaru, Luca Albertalli, Antonio Lioy, “The ForwardDiffSig Scheme for Multicast Authentication,” IEEE/ACM Transactions on Networking, Vol. 18, Iss. 6, Dec. 2010

J Park, H Kwon, N Kang, “IoT–Cloud collaboration to establish a secure connection for lightweight devices,” Springer Wireless Networks, Vol. 23, Iss. 3. April 2017.

Priyan Malarvizhi Kumar & Usha Devi Gandhi , “Enhanced DTLS with CoAP-based authentication scheme for the internet of things in healthcare application,” Springer, The Journal of Supercomputing. Vol. 76, 2020.

A. Perrig, R. Canetti, J. Tygar, D. Song, "Efficient and Secure Source Authentication for Multicast," in Proc. of the Network and Distributed System Security Symposium (NDSS), pp. 35-46, February 2001.

E. Sisinni, A. Saifullah, S. Han, U. Jennehag, and M. Gidlund, “Industrial internet of things: Challenges, opportunities, and directions,” IEEE Trans. Ind. Informat., vol. 14, no. 11, pp. 4724–4734, Nov. 2018.

T. Wang, H. Ke, X. Zheng, K. Wang, A. K. Sangaiah, and A. Liu, “Big data cleaning based on mobile edge computing in industrial sensor-cloud,” IEEE Trans. Ind. Informat., vol. 16, no. 2, pp. 1321–1329, Feb. 2020.

Q. Wu, F. Zhou, J. Xu, and Q. Wang, “Secure data stream outsourcing with publicly verifiable integrity in cloud storage,” J. Inf. Secur. Appl., vol. 49, 2019, Art. no. 102392.

Z. Zhang, X. Chen, J. Ma, and X. Tao, “New efficient constructions of verifiable data streaming with accountability,” Ann. Telecommun., vol. 74, no. 2, pp. 1–17, 2019.

S. Zhu, S. Xu, S. Setia, Z. Jajodia, “LHAP: a lightweight network access control protocol for ad-hoc networks,” Ad Hoc Networks Journal vol.4(5), pp. 567–585, May 2006.

B. Lu, U.W. Pooch, "A lightweight authentication protocol for mobile ad hoc networks," International Journal of Information Technology, vol11(2), pp. 119-135, February 2005.

R. Akbani, T. Korkmaz, G. Raju, "HEAP: A Packet Authentication Scheme for Mobile Ad hoc Networks," Ad Hoc Networks vol. 6(9), pp. 1134-1150, September 2008.

H. Kwon, J. Park, and N. Kang, “Challenges in Deploying CoAP over DTLS in Resource Constrained Environments,” in Proc. of WISA 2015, Aug. 2015