Journal of Multimedia Information System VOL. 8, NO. 1, March 2021 (pp. 57-60): ISSN 2383-7632 (Online)
http://doi.org/10.33851/JMIS.2021.8.1.57

Brief Paper:
Secret Key and Tag Generation for IIoT Systems Based on Edge Computing

Giheon Koh[1], Heungsik Yu[2], Sungun Kim[3*]

Abstract: Industry 4.0 is continuous automation by applying the latest smart technologies to traditional manufacturing industries. It means that large-scale M2M (Machine-to-Machine) communication and IoT (Internet of Things) technologies are well integrated to build efficient production systems by analyzing and diagnosing various issues without human intervention. Edge computing is widely used for M2M services that handle real-time interactions between devices at industrial machinery tool sites. Here, secure data transmission is required while interacting. Thus, this paper focused on a method of creating and maintaining secret key and security tag used for message authentication between end-devices and edge-device.

Key Words: Edge computing, Security tag, Secret key, IIoT.

I. INTRODUCTION

SMS (Smart Manufacturing System) is an application field of Industry 4.0 [1]. It is the digitization of all technologies related to manufacturing systems by providing inter-connectivity between physical entities (e.g. sensors, robots, machinery production tools) and cyber entities (e.g. network technology, big data technology, security technology, application technology). Additionally, this integrates IoT (Internet of Things) and M2M (Machine-to-Machine) technologies. As a result, it contributes to improving productivity through flexible manufacturing, real-time manufacturing control and monitoring, rapid response to market changes, application of smart sensor and robot technology, and big data analysis.

The IoT technology is the most important technology required to build such SMS. Especially, IIoT (Industrial Internet of Things) combines all physical and virtual entities to inter-operate through edge computing or cloud computing based on Internet [2]. Recently, cloud computing in the cyber domain is widely used to achieve SMS by controlling and managing the vast amounts of data generated from various IIoT end-devices in the physical domain, but it is not easy to accomplish the goal in most real-time cases.

On the other hand, edge computing can respond in real-time with a computing structure that helps in communicating, processing, storing, and operating data produced in the physical domain where end-devices are installed. That is, most of the functions are performed in the location where the end-devices are closed to. Thus, edge computing has the advantages of fast response time, real-time data analysis, non-essentiality of additional wide area network connections, and improved data security. Fig.1 shows a conceptual IIoT framework based on edge computing and cloud computing.

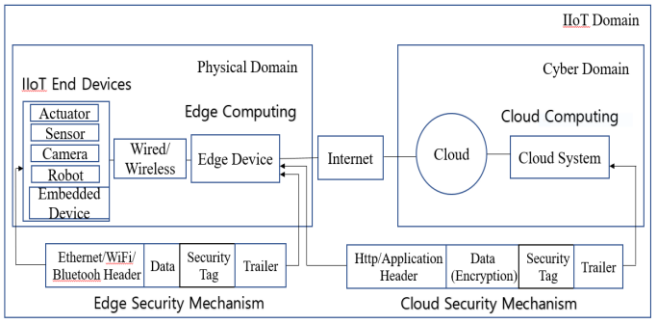

Fig. 1. IIoT framework based on edge computing and cloud computing.

Manuscript received February 25, 2021; Revised March 26, 2021; Accepted March 25, 2021. (ID No. JMIS-21M-03-007)
Corresponding Author (*): Sungun Kim, 45 Yongso-ro, Nam-Gu, Busan, Korea, +82-51-629-6235, kimsu@pknu.ac.kr.
[1]UNOMIC, Busan, South Korea, kion.koh@unomic.com
[2]UNOMIC, Busan, South Korea, paul.yu@unomic.com
[3]Department of Information and Communications Engineering, Pukyong National University, Busan, Korea, kimsu@pknu.ac.kr

In Fig. 1, the data generated from the end-devices must be delivered to the edge-device, or cloud system, in order to be processed and controlled. There are the risks of content leakage, forgery, and alteration due to hacking that could occur through other networks while transmitting data (here, content leakage indicates illegally used by an unauthorized third party, alteration indicates a form of content change, and forgery indicates when false data is intervened). In general, encryption technology could be used in response to content leakage. Security tag, as message authentication, is also useful to face tampering or forgery. In cloud computing, we need both technologies in a network environment. However, for edge computing in a short-range network environment, a security tag is sufficient to just cope with tampering and forgery. Here, the main issues are about how to create keys and tags with sufficient security capabilities with a low computational cost.

In this paper, we focused on the methodology to create security tags, generate and maintain security keys used for creating security tags. Furtherly, Chapter 2 describes related works, Chapter 3 explains how to create security tags and generate and maintain security keys, and Chapter 4 concludes our works.

II. RELATED WORKS

The data generated by IIoT end-devices is vast and it could be used for controlling and monitoring edge-devices. So that, because such data plays an important role, they must be protected from various threats.

As a block cipher, AES (Advanced Encryption Standard) is applied to data encryption and message authentication [3]. LEA (Lightweight Encryption Algorithm) is another block cipher with a 128-bit block [4]. The options for the key size are 128-bits, 192-bits, and 256-bits. It only consists of the ARX (modular Addition, bitwise Rotation, and bitwise XOR) operations for an array of four 32-bit words. LEA gets the resistance against the attacks for block ciphers and is proved by providing better speed and size on many platforms than AES [5]. For this reason, we chose the LEA approach to create a security tag for message authentication.

Esfahani et al. proposed a lightweight authentication mechanism using hash and XOR (eXclusive OR) operations for M2M communications in an IIoT environment [6]. M. Shin and S. Kim proposed a security mechanism for IoT services based on cloud and fog computing [7]. It is a variation of the WPA-2 (Wi-Fi Protected Access 2) method based on AES, and it is not smooth to change the key periodically which to enhance security. Even though the most of existing M2M communication protocols provide security mechanisms in an autonomous way, but it results in a high computational cost.

III. PROPOSED MECHANISM

3.1. Security Tag Generation

The process for security tag generation for the end-devices and edge-device explained in Fig. 2 is as follows. First, the data generated by the end-devices is divided by a multiple of 16-byte block. If data is not dividable with the multiple of 16-byte block, the final block is filled with zero-padding. Two inputs are required to apply the S-LEA (Simplified-LEA) method to generate the security tag. That is, they are the input data of multiple of 16-byte block and the 4-byte shared common secret key K generated by the end-devices and edge-device.

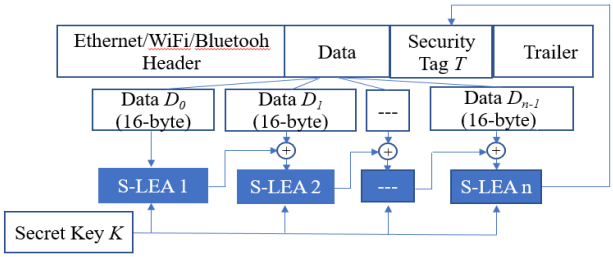

Fig. 2. Security tag generation for an edge security mechanism.

The procedure of security tag generation consists of n rounds for 32-bit key K. For n rounds, it encrypts a 128-bit data block $D_i = (D_i[0], D_i[1], D_i[2], D_i[3])$ to create a 128-bit security tag block $T = (T[0], T[1], T[2], T[3])$. Finally, the security tag T is produced from the obtained D_n after round iterations as shown in Fig. 3:

$T[0] = D_n[0]$, $T[1] = D_n[1]$, $T[2] = D_n[2]$, and $T[3] = D_n[3]$.

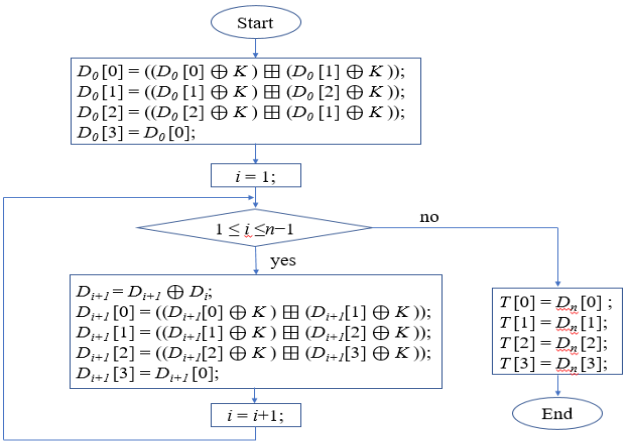

Fig. 3. Simplified LEA method for security tag generation.

Here, the notation $x \oplus y$ is defined as XOR of bit strings x and y with the same length. And $x \boxplus y$ is defined as addition modulo 2^{32} of 32-bit string x and y. Security was reinforced by using the result of the preceding S-LEA output block as input to the next step.

To shorten the calculation process while enhancing security, separate round keys are not used. Nevertheless, security capabilities could be supplemented by replacing the current key with a new key based on the period of time. Here, if the size of the input data block is small and if the speed has more weighted, the size of the data block should be modified into 4-byte or 8-byte.

3.2. Key Generation and Maintenance

To generate the secure key which is shared between the end-devices and edge-device, the Diffie-Hillman key generation method was adapted [8]. First, the end-devices and edge-device share two constants such as x (shared secret constant (bottom)) and y (shared secret constant (modulus)), and they have individual secret constants such as a (end-device's private secret constant) and b (edge-device's private secret constant). When generating a secret key initially, we use the shared secret constant (x, y) and private secret constant (a, b) owned. Then, the secret key constants (K_a: end-device secret key constant, K_b: edge-device secret key constant) are calculated through Equation (1).

$$K_a = x^a \% y,\ K_b = x^b \% y. \tag{1}$$

After exchanging the secret key constants, the first byte of the secret key is computed by equation (2). In this process, even if the end-device and edge-device use different private secret constants (a, b), but they generate the same value by the principle of discrete algebra.

$$key[0] = K_b{}^a \% y \text{ (End-device)} = K_a{}^b \% y \text{ (Edge-device)} \tag{2}$$

The second and third bytes are calculated as in Equation (3) and (4).

$$key[1] = ((K_a + K_b) \times key[0]) \% 255, \tag{3}$$

$$key[2] = ((K_a + K_b) \times key[1]) \% 255. \tag{4}$$

From the third to n^{th} bytes are obtained as in Equation (5).

$$key[n] = ((key[n\text{-}3] + key[n\text{-}2]) \times key[n\text{-}1]) \% 255. \tag{5}$$

Here, since $key[n]$ is newly obtained using the bytes of the previous two steps, safer key generation is guaranteed. As a result, the 4-byte block shared key K is combined from the finally obtained key in the following way:

$$K = [K[0] \leftarrow key[0]] \,\|\, [K[1] \leftarrow key[1]] \,\|\, [K[2] \leftarrow key[2]] \,\|\, [K[3] \leftarrow key[3]]. \tag{6}$$

We can also decrease or increase the size of the key to 2 bytes or 8 bytes. In addition, a new shared secret key is regenerated at regular intervals using a timer. If a timeout occurs while using the current key for the desired timer value, the value of the shared secret constant (x, y) is updated with the first and second bytes of the current key K being used, as shown in Equation (7):

$$x = K[0],\ y = K[1]. \tag{7}$$

Therefore, a new 32-bit block shared key K is generated by applying the result of Equation (7) to Equations (1), (2), (3), (4), (5), and (6).

3.3. Proposed Protocol

Fig. 4 and 5 explain sequence charts of the proposed protocol which are applied to create and maintain the secret key and security tag. Fig. 4 shows the case of normal transmission operation without being changed the contents by hackers.

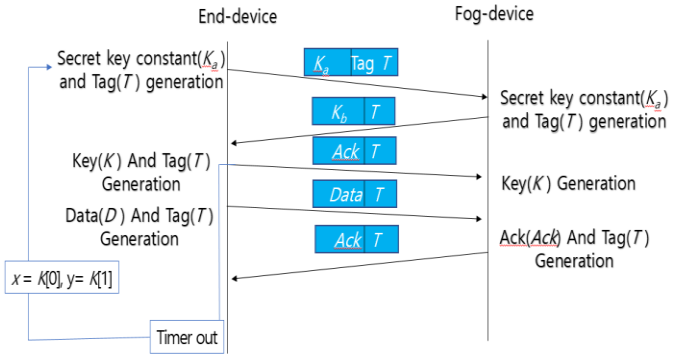

Fig. 4. Protocol in case of normal operation without data tampering or forgery.

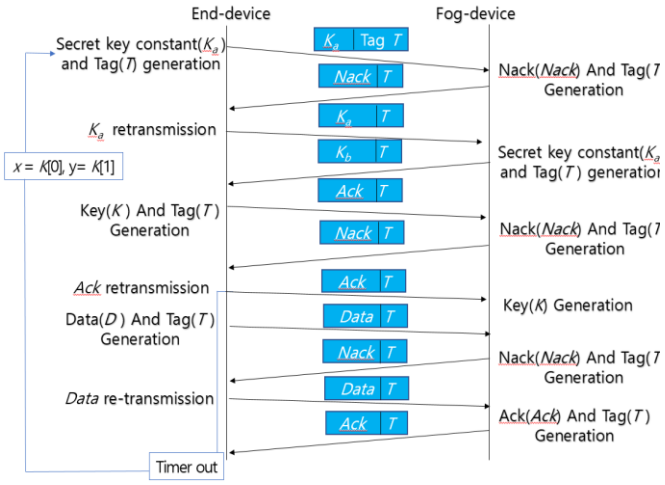

Fig. 5. Protocol in case of abnormal operation with data tampering or forgery.

Figure 5 shows a process in which the contents of transmitted data are changed by hackers or operated abnormally due to forgery or tampering, and so on. Ultimately, the proposed protocol is applied to ensure safe data transmission between the end-devices and edge-device.

IV. CONCLUSIONS

According to the works of D. Hong and J. Lee et Al. [4], LEA shows better performance. In our study, through implementing and verifying the proposed protocol for the end-devices and edge-device, it was confirmed that the same secret key was correctly generated, and the integrity of data was assured. And we verified that periodic key replacement and the shared secret constant update based on a timer are performed correctly. By doing this, the significant advantages are given in shortening the time to generate tags and enhancing the security capability by changing key periodically. We also experienced that there is no problem in using it in our test-environment based on an edge gateway system. Currently, efforts are being made to apply it in a real machine manufacturing system.

Acknowledgment

This research work was supported by the Research Grant of Pukyong National University (2019).

REFERENCES

[1] Kiwook Jung, Boonserm Kulvatunyou, Sangsu Choi, and Michael P. Brundage, "An Overview of a Smart Manufacturing System Readiness Assessment," *presented at the IFIP Advanced Information Technology,* vol. 488, pp. 705-712, Mar. 2017.

[2] David S. Linthicum, "Connecting Fog and Cloud Computing," *IEEE Cloud Computing*, vol. 4, no. 2, pp. 18-20, 2017.

[3] ADVANCED ENCRYPTION STANDARD (AES), Federal Information Processing Standards Publication 197, November 26, 2001.

[4] D. Hong and J. Lee, Dong-Chan Kim, Daesung Kwon, K. Ryu, D. Lee, "LEA: A 128-Bit Block Cipher for Fast Encryption on Common Processors," *presented at the 14th International Workshop on Information Security Applications*, Springer, vol. 8267, pp. 3-27, Aug. 2013.

[5] S. Lim, J. Lee, D. Han, "Improved Differential Fault Attack on LEA by Algebraic Representation of Modular Addition," *IEEE Access*, vol. 8, pp. 212794-212802, 2020.

[6] A. Esfahani, G. Mantas, R. Matischek, F. B. Saghezchi, et al., "A Lightweight Authentication Mechanism for M2M Communications in Industrial IoT Environment," *IEEE Internet of Things Journal*, vol. 6, no. 1, pp. 288-296, 2019.

[7] M. Shin and S. Kim, "A Study on the Security Framework for IoT Services based on Cloud and Fog Computing," *Journal of Korea Multimedia Society*, vol. 20, no. 12, pp. 3-27, Dec. 2017.

[8] IETF, "Diffie-Hellman Key Agreement Method," *RFC 2631*, 1999.